# TODo

在framework/base/core/java定义成另一个了

Settings.Secure.putInt(getContext().getContentResolver(), Settings.Secure.LOCATION\_MODE, mode);

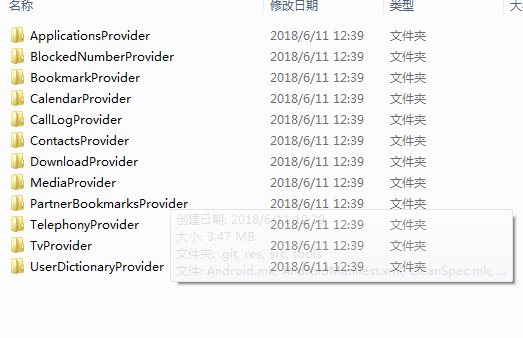
# Content Provider

Content Provider几乎已经成为了Android系统中两个或多个应用(系统应用和普通应用)之间共享数据库的最直接的方法。

在Android源码中，有两个目录是专门存放Content Provider。

一个是/源码目录/frameworks/providers，允许任何Adroid应用访问，包括：ApplicationProvider、CalendarProvider、ContactsProvider、DownloadProvider、MedialProvider、PartnerBookmarksProvider、TelephonyProvider、TvProvider、UserDictionaryProvider；

另一个是/源码目录/frameworks/base/packages，通常只允许系统级别的Android应用读写数据，**普通Android应用只能读数据**。包含：ExternalStorageProvider、SettingsProvider



## media数据库

*media相关的数据路径：   
data/data/com.android.providers.meida   
external.db是外置文件路径(SD卡存储的文件)   
internal.db是内存文件路径(手机存储的文件)*

# SettingsProvider

frameworks/base/packages/SettingsProvider/

Setting Content Provider 是所有系统设置的管理者。

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。

## 概述

### 主要源码

SettingsProvider的代码数量不多，主要包含如下的java文件：

frameworks/base/packages/SettingsProvider

DatabaseHelper.java

GenerationRegistry.java

SettingsHelper.java

SettingsProvider.java

SettingsState.java

frameworks/base/core/java/android/provider/Settings.java

### 比较

SettingsProvider顾名思义是一个提供设置数据共享的Provider，SettingsProvider和Android系统其它Provider有很多不一样的地方，如：

* SettingsProvider只接受int、float、string等基本类型的数据；
* SettingsProvider由Android系统framework进行了封装，使用更加快捷方便
* SettingsProvider的数据由键值对组成

SettingsProvider有点类似Android的properties系统（Android属性系统）：SystemProperties。SystemProperties除具有SettingsProvider以上的三个特性，SettingsProvider和SystemProperties的不同点在于：

* 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；
* 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；
* 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不可以使用。

### 存储设计

在M(Android5.0)版本之前，SettingsProvider中系统设置是存储在settings.db[数据库](http://lib.csdn.net/base/mysql)中；但是在L(Android6.0)之后，SettingsProvider中系统设置改为由xml存储在data分区。

Android M (6.0)之后的谷歌对SettingsProvider进行了重构，所以Settings数据库的路径有了变化 在手机目录/data/system/users/0路径下，以xml的形式存储数据，有settings\_global.xml, settings\_system.xml, settings\_secure.xml   
其中不同用户的数据库放不同的用户路径下，如果没有创建新用户，则在/data/system/users/0下。

在Android L版本中，Google团队为了优化系统性能、提高安全性等等一些列需求，重构了SettingsProvider模块，在L版本中，不在使用数据库来存储系统设置，而是通过xml将系统设置存储在了**/data/system/user/** 目录下



有关L版本中SettingsProvider的修改更新方法同之前的相同

## settings.db 详解

globl

？？？？？？？？？？？？？？

如Wifi状态，蓝牙是否开启，当前设备所处的模式等等，很多很多类似的全局属性控制的属性值。最重要的是global表，该表包含两个字段：name和value.其中name表示系统信息的key，value表示该系统设置项所对应的选项值。数据库结构如下：



### 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；
* 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。
* 目前来看区别不大
* Settings.db是系统的一个Provider，可以获取或者修改里面的数据值，这几个表都可以存储String,Int等类型的值，数据类型不受限制
* 系统可能会根据不同的数据进行分类
* 如果添加自己的数据，写在哪个表里都行，数据的获取和修改是需要的权限以及字段名的长度的限制
* 直接看代码

MOVED\_TO\_GLOBAL：<https://blog.csdn.net/u013398960/article/details/78857300>

字段复制

### settings\_ global.xml

<setting id="384" name="dji\_RECORDING" value="0" package="com.kpad.ui" />

<setting id="397" name="boot\_count" value="43" package="android" />

<setting id="4" name="airplane\_mode\_toggleable\_radios" value="bluetooth,wifi,nfc" package="android" />

<setting id="53" name="network\_scoring\_provisioned" value="1" package="android" />

<setting id="45" name="device\_name" value="rm500" package="android" />

<setting id="37" name="default\_install\_location" value="0" package="android" />

#### Keys

<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>

<settings version="132">

<setting id="42" name="low\_battery\_sound\_timeout" value="0" package="android" />

<setting id="33" name="car\_undock\_sound" value="/system/media/audio/ui/Undock.ogg" package="android" />

<setting id="171" name="enable\_cellular\_on\_boot" value="1" package="com.android.phone" />

<setting id="46" name="volte\_vt\_enabled" value="1" package="android" />

<setting id="893" name="development\_settings\_enabled" value="1" package="com.android.settings" />

<setting id="497" name="window\_animation\_scale" value="1.0" package="android" />

<setting id="635" name="bluetooth\_input\_device\_priority\_D8:C7:71:37:92:0D" value="-1" package="com.android.bluetooth" />

<setting id="453" name="mode\_ringer" value="2" package="android" />

<setting id="37" name="default\_install\_location" value="0" package="android" />

<setting id="817" name="wifi\_display\_on" value="1" package="com.android.settings" />

<setting id="45" name="device\_name" value="rm500" package="android" />

<setting id="660" name="wifi\_networks\_available\_notification\_on" value="0" package="com.android.settings" />

<setting id="29" name="dock\_sounds\_enabled\_when\_accessbility" value="0" package="android" />

<setting id="902" name="bluetooth\_on" value="1" package="android" />

<setting id="626" name="bugreport\_in\_power\_menu" value="0" package="com.android.settings" />

<setting id="18" name="mobile\_data" value="1" package="android" />

<setting id="38" name="emergency\_tone" value="0" package="android" />

<setting id="47" name="add\_users\_when\_locked" value="0" package="android" />

<setting id="35" name="dock\_audio\_media\_enabled" value="1" package="android" />

<setting id="929" name="adb\_enabled" value="1" package="com.android.settings" />

<setting id="25" name="trusted\_sound" value="/system/media/audio/ui/Trusted.ogg" package="android" />

<setting id="24" name="unlock\_sound" value="/system/media/audio/ui/Unlock.ogg" package="android" />

<setting id="3" name="airplane\_mode\_radios" value="cell,bluetooth,wifi,nfc,wimax" package="android" />

<setting id="21" name="wifi\_max\_dhcp\_retry\_count" value="9" package="android" />

<setting id="653" name="bluetooth\_input\_device\_priority\_9C:2E:A1:D8:31:02" value="-1" package="com.android.bluetooth" />

<setting id="4" name="airplane\_mode\_toggleable\_radios" value="bluetooth,wifi,nfc" package="android" />

<setting id="633" name="bluetooth\_input\_device\_priority\_EC:D0:9F:B0:DF:96" value="-1" package="com.android.bluetooth" />

<setting id="53" name="network\_scoring\_provisioned" value="1" package="android" />

<setting id="44" name="heads\_up\_notifications\_enabled" value="1" package="android" />

<setting id="624" name="bluetooth\_input\_device\_priority\_E4:46:DA:44:CF:3B" value="-1" package="com.android.bluetooth" />

<setting id="40" name="preferred\_network\_mode" value="9" package="android" />

<setting id="48" name="bluetooth\_disabled\_profiles" value="0" package="android" />

<setting id="31" name="desk\_undock\_sound" value="/system/media/audio/ui/Undock.ogg" package="android" />

<setting id="650" name="bluetooth\_a2dp\_sink\_priority\_60:60:1F:5E:1F:AA" value="-1" package="com.android.bluetooth" />

<setting id="493" name="mobile\_data\_always\_on" value="0" package="com.android.settings" />

<setting id="495" name="debug\_app" package="android" />

<setting id="51" name="webview\_provider" value="com.android.webview" package="android" />

<setting id="811" name="dji\_RECORDING" value="0" package="com.android.systemui" />

<setting id="9" name="wifi\_sleep\_policy" value="2" package="android" />

<setting id="28" name="dock\_sounds\_enabled" value="0" package="android" />

<setting id="816" name="charging\_sounds\_enabled" value="0" package="com.android.settings" />

<setting id="43" name="wifi\_scan\_always\_enabled" value="0" package="android" />

<setting id="501" name="overlay\_display\_devices" package="com.android.settings" />

<setting id="32" name="car\_dock\_sound" value="/system/media/audio/ui/Dock.ogg" package="android" />

<setting id="26" name="power\_sounds\_enabled" value="1" package="android" />

<setting id="8" name="stay\_on\_while\_plugged\_in" value="0" package="android" />

<setting id="931" name="zen\_mode\_config\_etag" value="1352430248" package="android" />

<setting id="36" name="set\_install\_location" value="0" package="android" />

<setting id="492" name="debug\_view\_attributes" value="0" package="com.android.settings" />

<setting id="49" name="zen\_mode" value="0" package="android" />

<setting id="500" name="transition\_animation\_scale" value="1.0" package="android" />

<setting id="636" name="bluetooth\_a2dp\_sink\_priority\_D8:C7:71:37:92:0D" value="-1" package="com.android.bluetooth" />

<setting id="170" name="cell\_on" value="1" package="com.android.phone" />

<setting id="16" name="data\_roaming" value="1" package="android" />

<setting id="11" name="package\_verifier\_enable" value="1" package="android" />

<setting id="2" name="theater\_mode\_on" value="0" package="android" />

<setting id="15" name="cdma\_cell\_broadcast\_sms" value="1" package="android" />

<setting id="168" name="airplane\_mode\_on" value="0" package="com.android.settings" />

<setting id="27" name="low\_battery\_sound" value="/system/media/audio/ui/LowBattery.ogg" package="android" />

<setting id="496" name="wait\_for\_debugger" value="0" package="android" />

<setting id="19" name="netstats\_enabled" value="1" package="android" />

<setting id="494" name="ota\_disable\_automatic\_update" value="1" package="com.android.settings" />

<setting id="41" name="subscription\_mode" value="0" package="android" />

<setting id="5" name="assisted\_gps\_enabled" value="1" package="android" />

<setting id="23" name="lock\_sound" value="/system/media/audio/ui/Lock.ogg" package="android" />

<setting id="475" name="auto\_time\_zone" value="0" package="com.android.settings" />

<setting id="922" name="wifi\_on" value="1" package="android" />

<setting id="34" name="wireless\_charging\_started\_sound" value="/system/media/audio/ui/WirelessChargingStarted.ogg" package="android" />

<setting id="776" name="device\_provisioned" value="1" package="com.dpad.launcher" />

<setting id="55" name="emergency\_affordance\_needed" value="0" package="android" />

<setting id="973" name="dji\_SRE\_LEVEL" value="3" package="com.android.systemui" />

<setting id="654" name="bluetooth\_a2dp\_sink\_priority\_9C:2E:A1:D8:31:02" value="-1" package="com.android.bluetooth" />

<setting id="634" name="bluetooth\_a2dp\_sink\_priority\_EC:D0:9F:B0:DF:96" value="-1" package="com.android.bluetooth" />

<setting id="59" name="sysui\_demo\_allowed" package="com.android.systemui" />

<setting id="625" name="bluetooth\_a2dp\_sink\_priority\_E4:46:DA:44:CF:3B" value="-1" package="com.android.bluetooth" />

<setting id="652" name="auto\_time" value="1" package="com.android.settings" />

<setting id="20" name="usb\_mass\_storage\_enabled" value="1" package="android" />

<setting id="649" name="bluetooth\_input\_device\_priority\_60:60:1F:5E:1F:AA" value="-1" package="com.android.bluetooth" />

<setting id="688" name="wifi\_p2p\_device\_name" value="rm500" package="android" />

<setting id="39" name="call\_auto\_retry" value="0" package="android" />

<setting id="62" name="audio\_safe\_volume\_state" value="3" package="android" />

<setting id="913" name="boot\_count" value="140" package="android" />

<setting id="30" name="desk\_dock\_sound" value="/system/media/audio/ui/Dock.ogg" package="android" />

<setting id="499" name="animator\_duration\_scale" value="1.0" package="android" />

### settings\_system.xml

#### hot

<setting id="30" name="dual\_screen\_icon\_used" value="0" package="android" />

<setting id="60" name="kji\_fn\_tip" value="1" package="com.android.settings" />

<setting id="12" name="screen\_off\_timeout" value="60000" package="android" />

<setting id="47" name="sre\_state" value="0" package="kji.system.launcher" />

<setting id="46" name="screen\_brightness" value="255" package="com.android.systemui" />

#### Key

<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>

<settings version="132">

<setting id="98" name="dji\_fn\_down" value="sre" package="com.android.settings" />

<setting id="47" name="dji\_fn\_left" value="wifi" package="com.android.settings" />

<setting id="18" name="window\_animation\_scale" value="1.0" package="android" />

<setting id="2312" name="volume\_music\_speaker" value="9" package="android" />

<setting id="22" name="notification\_light\_pulse" value="1" package="android" />

<setting id="100" name="dji\_fn\_right" value="sre" package="com.android.settings" />

<setting id="35" name="notification\_sound" value="content://media/internal/audio/media/98" package="com.android.providers.media" />

<setting id="30" name="dual\_screen\_icon\_used" value="0" package="android" />

<setting id="5" name="volume\_alarm" value="6" package="android" />

<setting id="1" name="volume\_music" value="11" package="android" />

<setting id="4" name="volume\_voice" value="4" package="android" />

<setting id="39" name="ringtone" value="content://media/internal/audio/media/180" package="com.android.providers.media" />

<setting id="14" name="hearing\_aid" value="0" package="android" />

<setting id="160" name="dji\_fn\_tip" value="1" package="com.android.settings" />

<setting id="2091" name="dji\_usb\_mode" value="auto" package="com.android.settings" />

<setting id="2" name="volume\_ring" value="5" package="android" />

<setting id="27" name="screenshot\_location" value="internal\_storage" package="android" />

<setting id="125" name="screen\_brightness\_mode" value="0" package="android" />

<setting id="34" name="alarm\_alert\_set" value="1" package="com.android.providers.media" />

<setting id="9" name="mute\_streams\_affected" value="46" package="android" />

<setting id="15" name="tty\_mode" value="0" package="android" />

<setting id="40" name="ringtone\_set" value="1" package="com.android.providers.media" />

<setting id="171" name="font\_scale" value="1.0" package="android" />

<setting id="11" name="dim\_screen" value="1" package="android" />

<setting id="36" name="notification\_sound\_set" value="1" package="com.android.providers.media" />

<setting id="7" name="volume\_bluetooth\_sco" value="7" package="android" />

<setting id="2329" name="lockscreen\_sounds\_enabled" value="1" package="com.android.settings" />

<setting id="20" name="accelerometer\_rotation" value="1" package="android" />

<setting id="12" name="screen\_off\_timeout" value="60000" package="android" />

<setting id="863" name="volume\_alarm\_speaker" value="2" package="android" />

<setting id="10" name="vibrate\_when\_ringing" value="0" package="android" />

<setting id="33" name="alarm\_alert" value="content://media/internal/audio/media/9" package="com.android.providers.media" />

<setting id="28" name="screenshot\_button\_show" value="1" package="android" />

<setting id="26" name="pointer\_speed" value="0" package="android" />

<setting id="19" name="transition\_animation\_scale" value="1.0" package="android" />

<setting id="31" name="mode\_ringer\_streams\_affected" value="166" package="android" />

<setting id="6" name="volume\_notification" value="5" package="android" />

<setting id="13" name="dtmf\_tone\_type" value="0" package="android" />

<setting id="23" name="dtmf\_tone" value="1" package="android" />

<setting id="29" name="dual\_screen\_mode" value="1" package="android" />

<setting id="914" name="volume\_ring\_speaker" value="2" package="android" />

<setting id="2292" name="system\_locales" value="en-GB,zh-Hans-CN,zh-Hant-HK,ja-JP" package="android" />

<setting id="37" name="volume\_music\_headset" value="10" package="android" />

<setting id="3" name="volume\_system" value="7" package="android" />

<setting id="101" name="dji\_fn\_c3" value="undefinded" package="com.android.settings" />

<setting id="2331" name="dji\_fn\_up" value="bluetooth" package="com.android.settings" />

<setting id="32" name="status\_bar\_show\_battery\_percent" package="com.android.systemui" />

<setting id="2358" name="dji\_SRE\_LEVEL" value="0" package="android" />

<setting id="2328" name="sound\_effects\_enabled" value="1" package="com.android.settings" />

<setting id="38" name="volume\_music\_headphone" value="10" package="android" />

<setting id="2338" name="screen\_brightness" value="115" package="com.android.systemui" />

<setting id="2330" name="haptic\_feedback\_enabled" value="1" package="com.android.settings" />

### cat settings\_secure.xml

<setting id="56" name="user\_setup\_complete" value="1" package="com.kpad.launcher" />

<setting id="11" name="long\_press\_timeout" value="400" package="android" />

<setting id="37" name="sysui\_tuner\_version" value="1" package="com.android.systemui" />

<setting id="19" name="screensaver\_components" value="com.google.android.deskclock/com.android.deskclock.Screensaver" package="android" />

<setting id="44" name="show\_note\_about\_notification\_hiding" value="0" package="com.android.systemui" />

<setting id="26" name="install\_non\_market\_apps" value="0" package="android" />

<setting id="33" name="show\_ime\_with\_hard\_keyboard" value="0" package="android" />

<setting id="35" name="multi\_press\_timeout" value="300" package="android" />

<setting id="51" name="enabled\_input\_methods" value="com.android.inputmethod.latin/.LatinIME" package="android" />

#### Input

/data/system/users/0 # cat settings\_secure.xml | grep input

<setting id="55" name="default\_input\_method" value="com.android.inputmethod.latin/.LatinIME" package="android" />

<setting id="64" name="input\_methods\_subtype\_history" value="com.android.inputmethod.latin/.LatinIME;-921088104" package="android" />

<setting id="72" name="selected\_input\_method\_subtype" value="-1" package="android" />

<setting id="46" name="selected\_spell\_checker" value="com.android.inputmethod.latin/.spellcheck.AndroidSpellCheckerService" package="android" />

<setting id="51" name="enabled\_input\_methods" value="com.android.inputmethod.latin/.LatinIME" package="android" />

#### Keys

<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>

<settings version="132">

<setting id="42" name="low\_battery\_sound\_timeout" value="0" package="android" />

<setting id="33" name="car\_undock\_sound" value="/system/media/audio/ui/Undock.ogg" package="android" />

<setting id="171" name="enable\_cellular\_on\_boot" value="1" package="com.android.phone" />

<setting id="46" name="volte\_vt\_enabled" value="1" package="android" />

<setting id="893" name="development\_settings\_enabled" value="1" package="com.android.settings" />

<setting id="497" name="window\_animation\_scale" value="1.0" package="android" />

<setting id="635" name="bluetooth\_input\_device\_priority\_D8:C7:71:37:92:0D" value="-1" package="com.android.bluetooth" />

<setting id="453" name="mode\_ringer" value="2" package="android" />

<setting id="37" name="default\_install\_location" value="0" package="android" />

<setting id="817" name="wifi\_display\_on" value="1" package="com.android.settings" />

<setting id="45" name="device\_name" value="rm500" package="android" />

<setting id="660" name="wifi\_networks\_available\_notification\_on" value="0" package="com.android.settings" />

<setting id="29" name="dock\_sounds\_enabled\_when\_accessbility" value="0" package="android" />

<setting id="902" name="bluetooth\_on" value="1" package="android" />

<setting id="626" name="bugreport\_in\_power\_menu" value="0" package="com.android.settings" />

<setting id="18" name="mobile\_data" value="1" package="android" />

<setting id="38" name="emergency\_tone" value="0" package="android" />

<setting id="47" name="add\_users\_when\_locked" value="0" package="android" />

<setting id="35" name="dock\_audio\_media\_enabled" value="1" package="android" />

<setting id="929" name="adb\_enabled" value="1" package="com.android.settings" />

<setting id="25" name="trusted\_sound" value="/system/media/audio/ui/Trusted.ogg" package="android" />

<setting id="24" name="unlock\_sound" value="/system/media/audio/ui/Unlock.ogg" package="android" />

<setting id="3" name="airplane\_mode\_radios" value="cell,bluetooth,wifi,nfc,wimax" package="android" />

<setting id="21" name="wifi\_max\_dhcp\_retry\_count" value="9" package="android" />

<setting id="653" name="bluetooth\_input\_device\_priority\_9C:2E:A1:D8:31:02" value="-1" package="com.android.bluetooth" />

<setting id="4" name="airplane\_mode\_toggleable\_radios" value="bluetooth,wifi,nfc" package="android" />

<setting id="633" name="bluetooth\_input\_device\_priority\_EC:D0:9F:B0:DF:96" value="-1" package="com.android.bluetooth" />

<setting id="53" name="network\_scoring\_provisioned" value="1" package="android" />

<setting id="44" name="heads\_up\_notifications\_enabled" value="1" package="android" />

<setting id="624" name="bluetooth\_input\_device\_priority\_E4:46:DA:44:CF:3B" value="-1" package="com.android.bluetooth" />

<setting id="40" name="preferred\_network\_mode" value="9" package="android" />

<setting id="48" name="bluetooth\_disabled\_profiles" value="0" package="android" />

<setting id="31" name="desk\_undock\_sound" value="/system/media/audio/ui/Undock.ogg" package="android" />

<setting id="650" name="bluetooth\_a2dp\_sink\_priority\_60:60:1F:5E:1F:AA" value="-1" package="com.android.bluetooth" />

<setting id="493" name="mobile\_data\_always\_on" value="0" package="com.android.settings" />

<setting id="495" name="debug\_app" package="android" />

<setting id="51" name="webview\_provider" value="com.android.webview" package="android" />

<setting id="811" name="dji\_RECORDING" value="0" package="com.android.systemui" />

<setting id="9" name="wifi\_sleep\_policy" value="2" package="android" />

<setting id="28" name="dock\_sounds\_enabled" value="0" package="android" />

<setting id="816" name="charging\_sounds\_enabled" value="0" package="com.android.settings" />

<setting id="43" name="wifi\_scan\_always\_enabled" value="0" package="android" />

<setting id="501" name="overlay\_display\_devices" package="com.android.settings" />

<setting id="32" name="car\_dock\_sound" value="/system/media/audio/ui/Dock.ogg" package="android" />

<setting id="26" name="power\_sounds\_enabled" value="1" package="android" />

<setting id="8" name="stay\_on\_while\_plugged\_in" value="0" package="android" />

<setting id="931" name="zen\_mode\_config\_etag" value="1352430248" package="android" />

<setting id="36" name="set\_install\_location" value="0" package="android" />

<setting id="492" name="debug\_view\_attributes" value="0" package="com.android.settings" />

<setting id="49" name="zen\_mode" value="0" package="android" />

<setting id="500" name="transition\_animation\_scale" value="1.0" package="android" />

<setting id="636" name="bluetooth\_a2dp\_sink\_priority\_D8:C7:71:37:92:0D" value="-1" package="com.android.bluetooth" />

<setting id="170" name="cell\_on" value="1" package="com.android.phone" />

<setting id="16" name="data\_roaming" value="1" package="android" />

<setting id="11" name="package\_verifier\_enable" value="1" package="android" />

<setting id="2" name="theater\_mode\_on" value="0" package="android" />

<setting id="15" name="cdma\_cell\_broadcast\_sms" value="1" package="android" />

<setting id="168" name="airplane\_mode\_on" value="0" package="com.android.settings" />

<setting id="27" name="low\_battery\_sound" value="/system/media/audio/ui/LowBattery.ogg" package="android" />

<setting id="496" name="wait\_for\_debugger" value="0" package="android" />

<setting id="19" name="netstats\_enabled" value="1" package="android" />

<setting id="494" name="ota\_disable\_automatic\_update" value="1" package="com.android.settings" />

<setting id="41" name="subscription\_mode" value="0" package="android" />

<setting id="5" name="assisted\_gps\_enabled" value="1" package="android" />

<setting id="23" name="lock\_sound" value="/system/media/audio/ui/Lock.ogg" package="android" />

<setting id="475" name="auto\_time\_zone" value="0" package="com.android.settings" />

<setting id="922" name="wifi\_on" value="1" package="android" />

<setting id="34" name="wireless\_charging\_started\_sound" value="/system/media/audio/ui/WirelessChargingStarted.ogg" package="android" />

<setting id="776" name="device\_provisioned" value="1" package="com.dpad.launcher" />

<setting id="55" name="emergency\_affordance\_needed" value="0" package="android" />

<setting id="973" name="dji\_SRE\_LEVEL" value="3" package="com.android.systemui" />

<setting id="654" name="bluetooth\_a2dp\_sink\_priority\_9C:2E:A1:D8:31:02" value="-1" package="com.android.bluetooth" />

<setting id="634" name="bluetooth\_a2dp\_sink\_priority\_EC:D0:9F:B0:DF:96" value="-1" package="com.android.bluetooth" />

<setting id="59" name="sysui\_demo\_allowed" package="com.android.systemui" />

<setting id="625" name="bluetooth\_a2dp\_sink\_priority\_E4:46:DA:44:CF:3B" value="-1" package="com.android.bluetooth" />

<setting id="652" name="auto\_time" value="1" package="com.android.settings" />

<setting id="20" name="usb\_mass\_storage\_enabled" value="1" package="android" />

<setting id="649" name="bluetooth\_input\_device\_priority\_60:60:1F:5E:1F:AA" value="-1" package="com.android.bluetooth" />

<setting id="688" name="wifi\_p2p\_device\_name" value="rm500" package="android" />

<setting id="39" name="call\_auto\_retry" value="0" package="android" />

<setting id="62" name="audio\_safe\_volume\_state" value="3" package="android" />

<setting id="913" name="boot\_count" value="140" package="android" />

<setting id="30" name="desk\_dock\_sound" value="/system/media/audio/ui/Dock.ogg" package="android" />

<setting id="499" name="animator\_duration\_scale" value="1.0" package="android" />

## manifest

|  |
| --- |
| <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.android.providers.settings"  coreApp="true"  android:sharedUserId="android.uid.system">  <application android:allowClearUserData="false"  android:label="@string/app\_label"  android:process="system"  android:backupAgent="SettingsBackupAgent"  android:killAfterRestore="false"  android:icon="@mipmap/ic\_launcher\_settings">    <!-- todo add: android:neverEncrypt="true" -->  <provider android:name="SettingsProvider" android:authorities="settings"  android:multiprocess="false"  android:exported="true"  android:writePermission="android.permission.WRITE\_SETTINGS"  android:singleUser="true"  android:initOrder="100" />  </application>  </manifest> |

上面的Manifest配置由sharedUserId可知，SettingsProvider运行在系统进程中，定义的ContentProvider实现类是SettingsProvider，Uri凭证是settings。

通过如上文件内容，我们大概可以得出几个结论：   
1. 拥有system用户权限。标签的coreApp属性是true,且android:sharedUserId=”android.uid.system”意味着SettingProvder与Settings同样拥有system用户的权限，且通过android:process=”system”可以看出，SettingProvider运行在**系统进程中**。   
2. provider标签中android:authorities=”settings”可以知道，访问SettingsProvider的Uri的开头部分一定是“content://settings”;

比如

CONTENT\_URI = Uri.parse("content://settings/global");

3. provider标签中android:writePermission=”android.permission.WRITE\_SETTINGS”可以知道，在执行写操作时需要指定WRITE\_SETTINGS权限。

## SettingsProvider的启动过程

启动SettingsProvider即运行SettingsProvider，和打开一个Activity类似，会回调ContentProvider的生命周期方法，首先的，会调用OnCreate()方法，如下：

### OnCreate

Dfsa

public boolean onCreate() {

synchronized (mLock) {

......

mHandlerThread = new HandlerThread(LOG\_TAG,

Process.THREAD\_PRIORITY\_BACKGROUND);

mHandlerThread.start();

mSettingsRegistry = new SettingsRegistry();

}

registerBroadcastReceivers();

startWatchingUserRestrictionChanges();

return true;

}

上面的代码首先是实例化一个HandlerThread的实例mHandlerThread，优先级为Process.THREAD\_PRIORITY\_BACKGROUND，下文会用到。

然后实例化SettingsRegistry的实例mSettingsRegistry，这一步很重要。

接着会注册广播接收器，所关心的广播包括设备用户变化以及APP卸载的广播，

设备用户的变化对大多数地方使用SettingProvider的影响不是很大，本文就不再阐述和用户变化相关的内容了。

但是APP卸载这里需要关注一下，当一个APP有数据保存在SettingsProvider时，APP被卸载后，被卸载的APP设置的所有数据都会被清除。：

### SettingsRegistry

回到SettingsRegistry的实例化过程，构造方法如下

public SettingsRegistry() {

......

migrateAllLegacySettingsIfNeeded();

}

迁移settings数据，迁移什么数据呢？从哪里迁移到哪里呢？继续往下看：

#### migrateAllLegacySettingsIfNeeded

d private void migrateAllLegacySettingsIfNeeded() {

synchronized (mLock) {

final int key = makeKey(SETTINGS\_TYPE\_GLOBAL, UserHandle.USER\_SYSTEM);

File globalFile = getSettingsFile(key);

if (globalFile.exists()) {

return;

}

......

DatabaseHelper dbHelper = new DatabaseHelper(getContext(), userId);

SQLiteDatabase database = dbHelper.getWritableDatabase();

migrateLegacySettingsForUserLocked(dbHelper, database, userId);

// Upgrade to the latest version.

UpgradeController upgrader = new UpgradeController(userId);

upgrader.upgradeIfNeededLocked();

......

}

}

上面的代码首先是调用了makeKey()方法，所谓makeKey()就是和上文中的数据分类小章节中提到的System、Global和Secure三种key。然后调用getSettingsFile()方法获取到一个File对象的实例，如下：

#### getSettingsFile

private File getSettingsFile(int key) {

if (isGlobalSettingsKey(key)) {

final int userId = getUserIdFromKey(key);

return new File(Environment.getUserSystemDirectory(userId),

SETTINGS\_FILE\_GLOBAL);

} else if (isSystemSettingsKey(key)) {

final int userId = getUserIdFromKey(key);

return new File(Environment.getUserSystemDirectory(userId),

SETTINGS\_FILE\_SYSTEM);

} else if (isSecureSettingsKey(key)) {

final int userId = getUserIdFromKey(key);

return new File(Environment.getUserSystemDirectory(userId),

SETTINGS\_FILE\_SECURE);

} else {

throw new IllegalArgumentException("Invalid settings key:" + key);

}

}

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/settings\_global.xml

/data/system/users/0/settings\_system.xml

/data/system/users/0/settings\_secure.xml

那么也就是说，Global类型的数据保存在文件settings\_global.xml中，System类型的数据保存在文件settings\_system.xml中，Secure类型的数据保存在文件settings\_secure.xml中。

#### migrateLegacySettingsForUserLocked

DatabaseHelper的onCreate()方法执行完毕后，这里又回到migrateAllLegacySettingsIfNeeded()方法中，DatabaseHelper创建完毕后，继续调用migrateLegacySettingsForUserLocked()方法，如下：

private void migrateLegacySettingsForUserLocked(DatabaseHelper dbHelper,

SQLiteDatabase database, int userId) {

// Move over the system settings.

final int systemKey = makeKey(SETTINGS\_TYPE\_SYSTEM, userId);

**ensureSettingsStateLocked**(systemKey);

SettingsState systemSettings = mSettingsStates.get(systemKey);

**migrateLegacySettingsLocked**(systemSettings, database, TABLE\_SYSTEM);

**systemSettings.persistSyncLocked();**

......

// Drop the database as now all is moved and persisted.

if (DROP\_DATABASE\_ON\_MIGRATION) {

dbHelper.dropDatabase();

} else {

dbHelper.backupDatabase();

}

}

上面的代码中的每个方法都是那么重要，首先是ensureSettingsStateLocked(systemKey)，如下：

##### ensureSettingsStateLocked

d

private void ensureSettingsStateLocked(int key) {

if (mSettingsStates.get(key) == null) {

......

SettingsState settingsState = new SettingsState(mLock, getSettingsFile(key), key,

maxBytesPerPackage, mHandlerThread.getLooper());

mSettingsStates.put(key, settingsState);

}

}

上面代码实例化一个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。

##### migrateLegacySettingsLocked

private void migrateLegacySettingsLocked(SettingsState settingsState,

SQLiteDatabase database, String table) {

SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();

queryBuilder.setTables(table);

Cursor cursor = queryBuilder.query(database, ALL\_COLUMNS,

null, null, null, null, null);

try {

......

while (!cursor.isAfterLast()) {

String name = cursor.getString(nameColumnIdx);

String value = cursor.getString(valueColumnIdx);

settingsState.insertSettingLocked(name, value,

SettingsState.SYSTEM\_PACKAGE\_NAME);

cursor.moveToNext();

}

} finally {

cursor.close();

}

}

上面这个方法，查询数据库中System所有的设置，然后在while循环中把每个值的信息作为insertSettingLocked()的参数，insertSettingLocked()方法如下：

###### insertSettingLocked

public boolean insertSettingLocked(String name, String value, String packageName) {

Setting oldState = mSettings.get(name);

String oldValue = (oldState != null) ? oldState.value : null;

if (oldState != null) {

......

} else {

Setting state = new Setting(name, value, packageName);

mSettings.put(name, state);

}

......

}

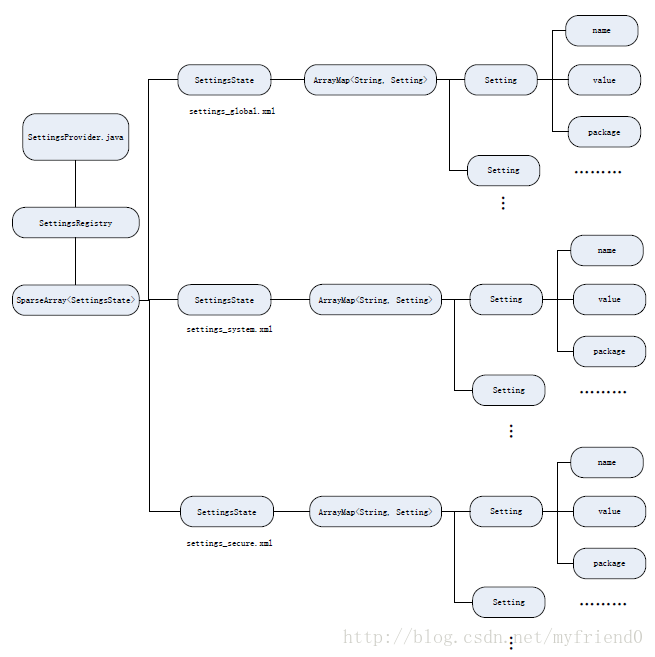
上面的方法把每个设置项封装到对象Setting中，接着有把state放置到ArrayMap<String, Setting>的实例mSettings中。

那么，从方法ensureSettingsStateLocked()到insertSettingLocked()方法，这个过程表明，有一个对象SettingsState，指向文件/data/system/users/0/settings\_system.xml，持有变量mSettings，而mSettings持有封装了设置项的name, value, packageName的对象Setting，换句话说，settings\_system.xml文件中的所有的设置项间接被SettingsState持有。

##### systemSettings.persistSyncLocked()，

又回到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，这个过程的代码就不贴出来了。migrateLegacySettingsForUserLocked()方法中省略的代码，就是和上文的这几个方法一样，把settings\_global.xml、settings\_secure.xml两个文件中的所有设置项封装到Setting中，被SettingsState持有。

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。它们的层次关系如下图：



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：

// Drop the database as now all is moved and persisted.

if (DROP\_DATABASE\_ON\_MIGRATION) {

dbHelper.dropDatabase();

} else {

dbHelper.backupDatabase();

}

如果是工程版本的系统，把数据库settings.db重命名为settings.db-backup，如果是非工程版本的系统，把数据库文件删除，也会删除日志settings.db-journal。

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。为什么会有这么一个过程，这些过程是否可以移除创建数据库这一步？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

权并注明出处。

### doWriteState

XmlSerializer serializer = Xml.newSerializer();

serializer.setOutput(out, StandardCharsets.UTF\_8.name());

serializer.setFeature("http://xmlpull.org/v1/doc/features.html#indent-output", true);

serializer.startDocument(null, true);

serializer.startTag(null, TAG\_SETTINGS);

serializer.attribute(null, ATTR\_VERSION, String.valueOf(version));

final int settingCount = settings.size();

for (int i = 0; i < settingCount; i++) {

Setting setting = settings.valueAt(i);

writeSingleSetting(mVersion, serializer, setting.getId(), setting.getName(),

setting.getValue(), setting.getPackageName());

if (DEBUG\_PERSISTENCE) {

Slog.i(LOG\_TAG, "[PERSISTED]" + setting.getName() + "=" + setting.getValue());

}

}

## 升级

public void upgradeIfNeededLocked() {

// The version of all settings for a user is the same (all users have secure).

SettingsState secureSettings = getSettingsLocked(

SETTINGS\_TYPE\_SECURE, mUserId);

// Try an update from the current state.

final int oldVersion = secureSettings.getVersionLocked();

final int newVersion = SETTINGS\_VERSION;

// If up do date - done.

if (oldVersion == newVersion) {

return;

}

onUpgradeLocked注意： SETTINGS\_VERSION 一定要更新，比如现在是currentVersion = 147，那你就要把SETTINGS\_VERSION改成147

## 操作SettingsProvider

由于Settings.java对使用SettingsProvider进行了封装，所以，使用起来相当简单简洁。由于Global、Secure、System三种数据类型的使用是几乎相同，所以本文就只以Global为例对查询插入数据的过程进行分析。

### 封装SettingsProvider接口

SettingsProvider是向整个Android系统提供用户编好设置的提供程序，所保存的数据类型和方式上也有一定约束和规定，且要求使用SettingsProvider是方便的，代码量少的。因此，需要对ContentProvider的一些接口进行封装，以保证在整个Android的java层任何一个地方都能方便、快捷的使用SettingsProvider进行数据查询，数据更新和数据插入。所以，理所当然地，framework有一个类Settings.java对使用SettingsProvider进行了封装。如下：

明出处。

这个类定义在文件frameworks/base/core/java/android/provider/Settings.java中。

public final class Settings {

public static final String AUTHORITY = "settings";

public static final class Global extends NameValueTable {

public static final Uri CONTENT\_URI = Uri.parse("content://" + AUTHORITY + "/global");

......

}

public static final class Secure extends NameValueTable {

public static final Uri CONTENT\_URI = Uri.parse("content://" + AUTHORITY + "/secure");

......

}

public static final class System extends NameValueTable {

public static final Uri CONTENT\_URI = Uri.parse("content://" + AUTHORITY + "/system");

......

}

private static class NameValueCache {

private final Uri mUri;

private final HashMap<String, String> mValues = new HashMap<String, String>();

public String getStringForUser(ContentResolver cr, String name, final int userHandle) {

......

}

public boolean putStringForUser(ContentResolver cr, String name, String value,

final int userHandle) {

......

}

private IContentProvider lazyGetProvider(ContentResolver cr) {

IContentProvider cp = null;

synchronized (NameValueCache.this) {

cp = mContentProvider;

if (cp == null) {

cp = mContentProvider = cr.acquireProvider(mUri.getAuthority());

}

}

return cp;

}

}

上面的代码中，分别声明了Global、Secure、System三个静态内部类，分别对应SettingsProvider中的Global、Secure、System三种数据类型。Global、Secure、System三个静态内部类会分别持有自己NameValueCache的实例变量，每个NameValueCache持有指向SettingsProvider中的SettingsProvider.java的AIDL远程调用IContentProvider，读者可以阅读《Android System Server大纲之ContentService和ContentProvider原理剖析》了解ConatentProvider的这个过程。因此，查询数据需要经过NameValueCache的getStringForUser()方法，插入数据需要经过putStringForUser()方法。同时，NameValueCache还持有一个变量mValues，用于保存查询过的设置项，以便下下次再次发起查询时，能够快速返回。

### Get

frameworks/base/core/java/android/provider/Settings.java中

#### getString

从SettingsProvider的Global中查询数据，查询是否是飞行模式使用方法如下：

String globalValue = Settings.Global.getString(getContentResolver(), Settings.Global.AIRPLANE\_MODE\_ON);

上面的代码，用起来代码量很少，只需要一行代码即可查询到所需要的值。深入Settings.java看getString()方法：

public static String getString(ContentResolver resolver, String name) {

return getStringForUser(resolver, name, UserHandle.myUserId());

}

/\*\* @hide \*/

public static String getStringForUser(ContentResolver resolver, String name,

int userHandle) {

if (MOVED\_TO\_SECURE.contains(name)) {

Log.w(TAG, "Setting " + name + " has moved from android.provider.Settings.Global"

+ " to android.provider.Settings.Secure, returning read-only value.");

return Secure.getStringForUser(resolver, name, userHandle);

}

return sNameValueCache.getStringForUser(resolver, name, userHandle);

}

getString()直接调用了getStringForUser()，getStringForUser()首先有做一个判断MOVED\_TO\_SECURE.contains(name)，做这个判断是因为在Android系统的更新中，保存在Global、Secure、System三种类型的数据的存放位置有变化，所以需要加这个判断兼容老版本的使用方法。在章节“封装SettingsProvider接口”中提到，查询必须经过NameValueCache.getStringForUser()方法，如下：

#### getStringForUser

public String getStringForUser(ContentResolver cr, String name, final int userHandle) {

final boolean isSelf = (userHandle == UserHandle.myUserId());

if (isSelf) {

......

} else if (mValues.containsKey(name)) {

return mValues.get(name);

......

IContentProvider cp = lazyGetProvider(cr);

// Try the fast path first, not using query(). If this

// fails (alternate Settings provider that doesn't support

// this interface?) then we fall back to the query/table

// interface.

if (mCallGetCommand != null) {

try {

......

Bundle b = cp.call(cr.getPackageName(), mCallGetCommand, name, args);

if (b != null) {

String value = b.getString(Settings.NameValueTable.VALUE);

......

mValues.put(name, value);

}

return value;

}

} catch (RemoteException e) {

// Not supported by the remote side? Fall through

// to query().

}

}

Cursor c = null;

try {

c = cp.query(cr.getPackageName(), mUri, SELECT\_VALUE, NAME\_EQ\_PLACEHOLDER,

new String[]{name}, null, null);

String value = c.moveToNext() ? c.getString(0) : null;

synchronized (NameValueCache.this) {

mValues.put(name, value);

}

return value;

......

if (c != null) c.close();

}

}

}

首先从缓存mValues变量中去找，如果没有查询到，就调用SettingsProvider的call()接口，如果call()接口也没有查询到，再调用query()接口。这里用的是call()接口，下文就以call()接口往下分析。cp.call()会调用到SettingsProvider的call()方法，读者可以阅读《Android System Server大纲之[ContentService和ContentProvider原理剖析](https://link.jianshu.com/?t=http://blog.csdn.net/myfriend0/article/details/58587065)》了解ConatentProvider的这个过程。注意参数mCallGetCommand。

#### SettingsProvider. call

D

public Bundle call(String method, String name, Bundle args) {

final int requestingUserId = getRequestingUserId(args);

switch (method) {

case Settings.CALL\_METHOD\_GET\_GLOBAL: {

Setting setting = **getGlobalSetting**(name);

return packageValueForCallResult(setting, isTrackingGeneration(args));

}

case Settings.CALL\_METHOD\_GET\_SECURE: {

Setting setting = getSecureSetting(name, requestingUserId);

return packageValueForCallResult(setting, isTrackingGeneration(args));

}

......

}

return null;

}

上层传过来的参数的command是mCallGetCommand，即Settings.CALL\_METHOD\_GET\_GLOBAL，所以调用getGlobalSetting()，方法，在章节“SettingsProvider的启动过程”中可知，getGlobalSetting()返回的Setting setting是封装了设置项name、value等信息的，查看getGlobalSetting()方法：

##### getGlobalSetting

private Setting getGlobalSetting(String name) {

// Get the value.

synchronized (mLock) {

return mSettingsRegistry.getSettingLocked(SETTINGS\_TYPE\_GLOBAL,

UserHandle.USER\_SYSTEM, name);

}

}

##### mSettingsRegistry.getSettingLocked()

public Setting getSettingLocked(int type, int userId, String name) {

final int key = makeKey(type, userId);

SettingsState settingsState = peekSettingsStateLocked(key);

return settingsState.getSettingLocked(name);

}

##### peekSettingsStateLocked

private SettingsState peekSettingsStateLocked(int key) {

SettingsState settingsState = mSettingsStates.get(key);

if (settingsState != null) {

return **settingsState**;

}

ensureSettingsForUserLocked(getUserIdFromKey(key));

return mSettingsStates.get(key);

}

获取到SettingsState settingsState，回到getSettingLocked()，调用settingsState.getSettingLocked(name)：

public Setting getSettingLocked(String name) {

if (TextUtils.isEmpty(name)) {

return mNullSetting;

}

Setting setting = mSettings.get(name);

if (setting != null) {

return new Setting(setting);

}

return mNullSetting;

}

到getSettingLocked()最终获取到Setting setting，其实这个过程就是章节“SettingsProvider的启动过程”中的层次关系图的反映，读者可以查看这个图更好理解这个过程。

### 插入数据

作者：FamilyYuan

<https://www.jianshu.com/p/d48977f220ee>

### 第三方APP使用SettingsProvider

第三方APP可以通过framework的Settings.java查询SettingsProvider中的设置项，使用方法查阅章节“查询数据”。第三APP是否可以修改SettingsProvider的设置项？Android系统不允许第三方APP修改SettingsProvider中的设置项。

权限问题

查阅SettingsProvider的设置项不需要声明任何权限。

修改SettingsProvider需要权限：

android.permission.WRITE\_SETTINGS，Protection level: signature

Secure数据：android.permission.WRITE\_SECURE\_SETTINGS，Not for use by third-party applications.

对已Global和Secure模块，还需要关心上文中的isGlobalOrSecureSettingRestrictedForUser()方法设置到的限制。

### And7插入流程分析

<https://blog.csdn.net/tangzhihai0421/article/details/71637287>

[**测试**](http://lib.csdn.net/base/softwaretest)的手机上是/data/system/users/0，该目录的settings\_global.xml，settings\_secure.xml和settings\_system.xml三个xml文件就是SettingsProvider中的数据文件

从getSettingsFile中可以看到SettingsProvider三个表global，system和secure的文件，它们的路径是相同的，通过Environment.getUserSystemDirectory确立

frameworks/base/core/java/android/os/Environment.java

**[java]** [view plain](http://blog.csdn.net/firedancer0089/article/details/70683085) [copy](http://blog.csdn.net/firedancer0089/article/details/70683085)

[在CODE上查看代码片](https://code.csdn.net/snippets/2354289)

1. **public** **static** File getUserSystemDirectory(**int** userId) {
2. **return** **new** File(**new** File(getSystemSecureDirectory(), "users"), Integer.toString(userId));
3. }

**[java]** [view plain](http://blog.csdn.net/firedancer0089/article/details/70683085) [copy](http://blog.csdn.net/firedancer0089/article/details/70683085)

[在CODE上查看代码片](https://code.csdn.net/snippets/2354289)

1. **public** **static** File getSystemSecureDirectory() {
2. **if** (isEncryptedFilesystemEnabled()) { //加密文件系统分支，默认是关闭的
3. **return** **new** File(SECURE\_DATA\_DIRECTORY, "system");
4. } **else** {
5. **return** **new** File(DATA\_DIRECTORY, "system");
6. }
7. }

**[java]** [view plain](http://blog.csdn.net/firedancer0089/article/details/70683085) [copy](http://blog.csdn.net/firedancer0089/article/details/70683085)

[在CODE上查看代码片](https://code.csdn.net/snippets/2354289)

1. **private** **static** **final** File DATA\_DIRECTORY
2. = getDirectory("ANDROID\_DATA", "/data");

依据ANDROID\_DATA环境变量获取路径，获取不到的话就默认为/data。

获取到DATA\_DIRECTORY后，后续还要添加"/ " + "system" + "/" + "users" + "/" + userId的值

例如我[**测试**](http://lib.csdn.net/base/softwaretest)的手机上是/data/system/users/0，该目录的settings\_global.xml，settings\_secure.xml和settings\_system.xml三个xml文件就是SettingsProvider中的数据文件。

这里虽然手机目录中数据库存储是以xml的形式，但是代码中操作时操作的仍然是数据库。系统会先生成Settings.db数据库，然后将其中的内容保存到xml文件中，最后删除Settings.db数据库(SettingsProvider的功能，后续会讲到)。   
这里为何先生成数据库后又删除，主要是为了兼容之前的版本。   
为什么以xml的形式保存字段：节省资源，简单，安全(xml保存的路径相对于之前的数据库路径比较安全，大概是考虑到多用户吧)

## 修改db

### 查看工具

#### Navicat推荐

#### sqlite3

# cd /data/data/com.android.providers.settings/databases

# sqlite3 settings.db

sqlite3 有可能提示找不到，这是由于在当前版本中sqlite3程序没有被编译到system中，这里我们只需要手动push一个sqlite3就好了。Cs果然没有，android 7是已经有了

在使用sqlite3打开settings.db数据库之后，可以使用.help来查看有关sqlite3的一些帮助信息。这里常用的如下所示：



### 命令settings put/get

settings get system ringtone

settings get global ringtone

root@gl300k:/ # settings put secure icon\_blacklist ethernet

root@gl300k:/ # settings get secure icon\_blacklist

ethernet

修改数据库中的字段使用命令   
adb shell settings get global captive\_portal\_server   
adb shell settings put global captive\_portal\_server 0

注：这里虽然手机目录中数据库存储是以xml的形式，但是代码中操作时操作的仍然是数据库。系统会先生成Settings.db数据库，然后将其中的内容保存到xml*文件中，最后删除Settings.db数据库(SettingsProvider的功能，后续会讲到)。   
这里为何先生成数据库后又删除，主要是为了兼容之前的版本。   
为什么以xml的形式保存字段：节省资源，简单，安全(xml保存的路径相对于之前的数据库路径比较安全，大概是考虑到多用户吧)*

**adb shell settings**,有关其详细用法，可以参照如下提示：

$ adb shell settings get system ringtone

content://media/internal/audio/media/59

$ adb shell settings set system ringtone "content://media/internal/audio/media/60"

Invalid command: set

usage: settings [--user NUM] get namespace key

settings [--user NUM] put namespace key value

settings [--user NUM] delete namespace key

'namespace' is one of {system, secure, global}, case-insensitive

If '--user NUM' is not given, the operations are performed on the owner user.

$ adb shell settings put system ringtone "content://media/internal/audio/media/60"

$ adb shell settings get system ringtone

content://media/internal/audio/media/60

### App访问

无法在普通app上通过以下方式使用，即使申请了android.permission.WRITE\_SETTINGS等权限也不行。会报SecurityException。

Settings.Global.*putString*(getContentResolver(),  
 **"myKey"**,**"v1"**);  
String v1 = Settings.Global.*getString*(getContentResolver(),  
 **"myKey"**);

### 定制默认值

如果要在系统中添加默认设置，需要如下步骤（以带物理键盘的系统中默认打开软键盘为例）：

1、在defaults.xml中为新加的数据定义一个默认值

<integer name="show\_ime\_with\_hard\_keyboard">1</integer>

2、在DatabaseHelper.java中的loadSecureSettings(SQLiteDatabase db)方法中加入新字段的添加代码

loadStringSetting(stmt, Settings.Secure.SHOW\_IME\_WITH\_HARD\_KEYBOARD,

R.string.show\_ime\_with\_hard\_keyboar);

3、重新编译

## 总结

本文从SettingsProvider的启动过程到使用SettingsProvider查询插入数据进行了详细的过程描述，在SettingsProvider的启动过程中，需要创建数据库，把默认设置项值写入到数据库，把数据中的所有数据，迁移到xml文件中。SettingsProvider的查询和插入结合章节“SettingsProvider的启动过程”中的层次关系图，一目了然。对于第三方APP只有读没有写的能力。由于SettingsProvider的特性，虽然SettingsProvider是跨进程通信，但是由于从多个层次都做了缓存，且SettingsProvider中的同步协作机制，只要时间都是花费在Binder通信上面，但是Binder通信是一种快速的跨进程通信的过程，所以在主线程（UI线程）中可以直接使用SettingsProvider查阅插入数据而不会导致UI阻塞导致ANR（应用程序无响应）。另外，由于SettingsProvider的特性和限制，SettingsProvider不予写入过多的数据，最好只是系统设置相关的设置项才保存到SettingsProvider中，同时也不适合写入过大的数据，否则将会严重影响SettingsProvider的性能。

权并注明出处。

## Qita Src

在SettingsProvider的src源代码中，共有四个类：   
1. DatabaseHelper类。数据库操作类，实现对setting.db的读写功能，继承自SQLiteOpenHelper类   
2. SettingsBackupAgent类。用于备份和恢复系统设置等功能的类，继承自BackupAgentHelper类   
3. SettingsHelper类。Settings的帮助类   
4. SettingsProvider类。继承自ContentProvider类的最核心的文件   
在这四个类中，最最重要的就是SettingsProvider，其次我关心的可能就是DatabaseHelper类是如何执行数据库语句的。

#### DatabaseHelper

上文中的migrateAllLegacySettingsIfNeeded()方法，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。从Android SQLite的架构可知，这个过程会调用SQLiteOpenHelper的onCreate()方法，如果读者对这个过程迷惑的，可以阅读Android的API指南Android SQLite API 指南。

首先，来分析DatabaseHelper类，在DatabaseHelper类中，就是对数据库及表的操作。主要是表的创建，数据库版本更新。onCreate方法中负责创建表，执行表创建语句，然后进行表的初始化，也就是系统的初始设置；onUpgrade方法最长，需要从第一个版本到最新版本一个一个判断，升级。我们看一下具体的初始设置：

|  |
| --- |
| //onCreate中的初始化设置调用  // Load initial volume levels into DB  loadVolumeLevels(db);  // Load inital settings values  loadSettings(db);  //下面是loadSettings方法：  private void loadSettings(SQLiteDatabase db) {  loadSystemSettings(db);  loadSecureSettings(db);  // The global table only exists for the 'owner' user  if (mUserHandle == UserHandle.USER\_OWNER) {  loadGlobalSettings(db);  }  }  //我们选择loadGlobalSettings进行查看，下面分别是设置boolean值，String值，int值  // --- Previously in 'system'  loadBooleanSetting(stmt, Settings.Global.AIRPLANE\_MODE\_ON,  R.bool.def\_airplane\_mode\_on);  loadStringSetting(stmt, Settings.Global.AIRPLANE\_MODE\_RADIOS,  R.string.def\_airplane\_mode\_radios);  loadIntegerSetting(stmt, Settings.Global.WIFI\_SLEEP\_POLICY,  R.integer.def\_wifi\_sleep\_policy); |

通过上面的代码我们看到，最后其实就是引用（或者说是读取）了integer或者bool再或者string中的一些具体的字段，设置到数据库中。我们要想查看这些默认的值，可以到res/values中进行查看。和本地化无关的存放在res/values/default.xml文件中，和本地化有关的（主要指string,涉及语言）则放在相应的语言分包下面；另外，我们还能注意到的是，sqlite不支持boolean值，如果要保存boolean值，是通过integer来保存的。一般1为true,0为false。

#### SettingsProvider

看onCreate（）方法，创建表和各表的索引，然后调用了loadBookmarks(db); loadVolumeLevels(db); loadSettings(db); 三个方法，loadBookmarks(db);是存系统几个常用的应用的intent ( 如，通讯录app，emailApp, 短信app,)，loadVolumeLevels(db); 初始化的是与声音相关的配制， loadSettings(db);方法就是从defaults.xml文件中读取默认值存入相应的表中，验证了之前 的猜测。

loadSettings(db); 里面的三个方法者是从defaults.xml文件中读取默认值存入相应的表中，loadSystemSettings(db)是存入System表中，loadSecureSettings(db)是存入 secure表中。

下面来分析最最核心的SettingsProvider的代码，核心就是insert,delete,update,query四个方法，我们依次看四个方法：

insert方法：insert方法中调用了insertForUser(Uri url, ContentValues initialValues, int desiredUserHandle) 方法。在insert方法中在执行insert方法之前，调用了检查写权限的方法checkWritePermissions，更新db都需要android.Manifest.permission.WRITE\_SECURE\_SETTINGS权限（需要系统签名）和android.Manifest.permission.WRITE\_、SETTINGS。读取则不需要任何权限。

|  |
| --- |
| **private void** checkWritePermissions(SqlArguments args) {  **if** ((TABLE\_SECURE.equals(args.table) || TABLE\_GLOBAL.equals(args.table)) &&  getContext().checkCallingOrSelfPermission(  android.Manifest.permission.WRITE\_SECURE\_SETTINGS) !=  PackageManager.PERMISSION\_GRANTED) {  **throw new** SecurityException(  String.format(**"Permission denial: writing to secure settings requires %1$s"**,  android.Manifest.permission.WRITE\_SECURE\_SETTINGS));  } } |

delete和update方法中也调用了如上的权限检查方法，然后再执行相应的数据库操作方法

query方法：query方法中调用了queryForUser()方法进而执行query查询返回Cursor结果对象。

## 常见问题

主意：

正常的

root@zs600b:/system/priv-app # ls -al | grep Set

drwxr-xr-x root root 2017-07-06 11:32 Settings

drwxr-xr-x root root 2017-07-06 11:30 SettingsProvider

异常：

root@zs600b:/system/priv-app # ls -al | grep Set

drwxr-xr-x root root 2017-08-29 21:42 Settings

drwxr-xr-x root root 2017-08-29 21:40 SettingsProvider

-rw-r--r-- root root 126118 2017-08-31 21:35 SettingsProvider.apk

Log：

09-01 10:10:20.509 I/PackageManager( 486): Package com.android.providers.settings at /system/priv-app/SettingsProvider.apk ignored: updated version 22 better than this 22

09-01 10:10:20.509 W/PackageManager( 486): Code path for hidden system pkg : com.android.providers.settings changing from /system/priv-app/SettingsProvider to /system/priv-app/SettingsProvider.apk

09-01 10:10:20.509 W/PackageManager( 486): Failed to parse /system/priv-app/SettingsProvider.apk: null

root@zs600b:/system/priv-app/SettingsProvider/arm # ls -al

-rw-r--r-- root root 254440 2017-08-29 21:40 SettingsProvider.odex

09-01 10:47:21.478 W/art ( 489): Failed to find OatDexFile for DexFile /system/priv-app/SettingsProvider/SettingsProvider.apk ( canonical path /system/priv-app/SettingsProvider/SettingsProvider.apk) with checksum 0x6ade6c23 in OatFile /system/priv-app/SettingsProvider/arm/SettingsProvider.odex

09-01 10:47:21.504 I/dex2oat ( 549): Error opening the directory: /system/lib/plugins

09-01 10:47:21.505 I/dex2oat ( 549): /system/bin/dex2oat --runtime-arg -classpath --runtime-arg --instruction-set=arm --instruction-set-features=div --runtime-arg -Xrelocate --boot-image=/system/framework/boot.art --dex-file=/system/priv-app/SettingsProvider/SettingsProvider.apk --oat-fd=76 --oat-location=/data/dalvik-cache/arm/system@priv-app@SettingsProvider@SettingsProvider.apk@classes.dex --runtime-arg -Xms64m --runtime-arg -Xmx512m

adb install –r

INSTALL\_FAILED\_ALREADY\_EXISTS

means you uninstall before install -r remove purpose

## 历史版本

### Android M之前的SettingsProvider

**frameworks/base/packages/SettingsProvider/res/values/defaults.xml**

**如何添加属性：**系统属性的设置，我们可以通过修改源码，编译生成SettingsProvider.apk并安装到系统中，就能生效。

## Ref

[**Android SettingProvider详解**](http://blog.csdn.net/peng_cao/article/details/50887509)

[Settings.Global](https://developer.android.com/reference/android/provider/Settings.Global.html)

<https://www.jianshu.com/p/59302a400944>

Android 系统提供媒体库 URI 与 数据库的对应关系

[Android系统APP之SettingsProvider](https://www.jianshu.com/p/d48977f220ee)

# 其他db

Gs

Ctrl+Z推出命令模式

rm500:/data/system\_de/0 # sqlite3 accounts\_de.db

sqlite> .databases

seq name file

--- --------------- ----------------------------------

0 main /data/system\_de/0/accounts\_de.db

sqlite> .dump shared\_accounts

PRAGMA foreign\_keys=OFF;

BEGIN TRANSACTION;

CREATE TABLE shared\_accounts ( \_id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, type TEXT NOT NULL, UNIQUE(name,type));

COMMIT;

sqlite> .show

echo: off

eqp: off

explain: off

headers: off

mode: list

nullvalue: ""

output: stdout

colseparator: "|"

rowseparator: "\n"

stats: off

width:

**sqlite> select \* from accounts;**

[7|key.guan@kji.com|com.kji.oo||1535041382744](mailto:7|key.guan@kji.com|com.kji.oo||1535041382744)

就一个账户名字，password呢，收藏这些数据怎么来的？app2再登陆一次？？