目 录

文档修订记录 2

目 录 3

1 简介 4

1.1 文档目的 4

1.2 使用范围 4

2 简介 4

2.1 短按power键流程 5

2.1.1 Android4.X标准短按power键处理流程 5

2.1.2 amlogic短按power键处理流程 7

2.1 长按power键流程 8

2.1.1 Android4.X标准长按power键处理流程 8

2.1.2 Amlogic长按power键处理流程 9

2.1.3 Message超时处理流程 9

3 如何处理短按和长按电源键 11

# 简介

## 文档目的

Android 的电源管理也是很重要的一部分。比如在待机的时候关掉不用的设备，timeout之后的屏幕和键盘背光的关闭，用户操作的时候该打开多少设备等等，这些都直接关系到产品的待机时间，以及用户体验。

了解清楚Android系统Power键处理流程,区分出长按与短按power键的区别,还有长按power键如何弹出提示框. (主要目的是目前弹的是系统内部的框，需求我们要弹出第三方具有对Power按键处理能力的应用)

## 使用范围

Android 4.0和4.0以上系统对Power键处理的相关问题。

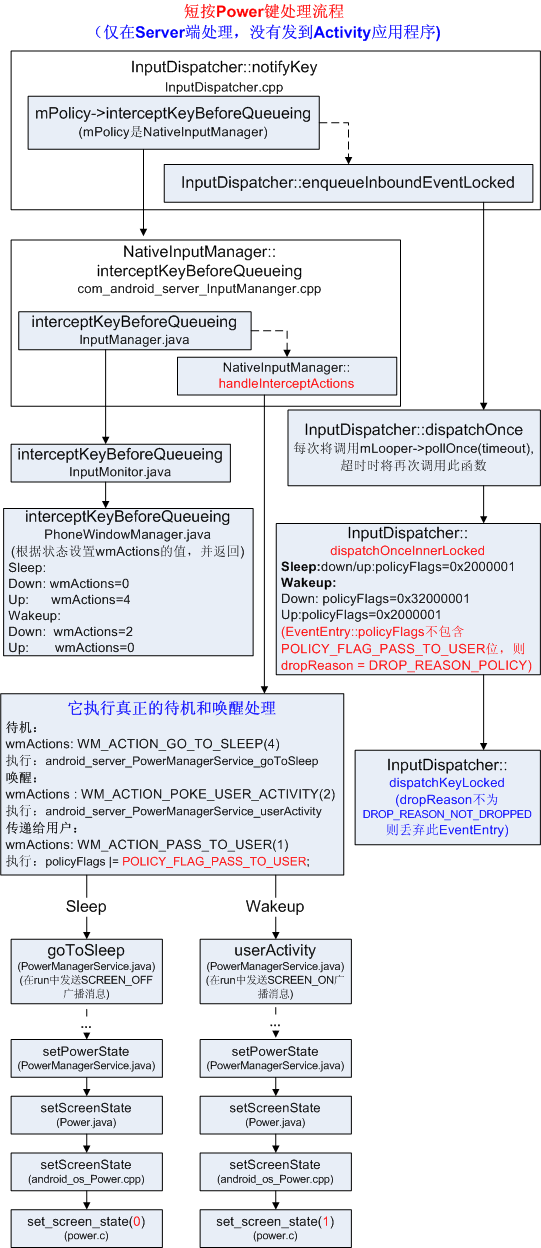
# 简介

 Android4.x在Framework的PhoneWindowManager对Power（KeyEvent.KEYCODE\_POWER）和Home（KeyEvent.KEYCODE\_HOME）键做了处理，不会把这些键传送上层应用程序。如需要把这些键发送给Activity和Service，需要在PhoneWindowManager处理这些键时“发送一个广播出去，然后在应用程序接收到广播后做处理”。

       如果应用程序只需要获取获取待机、唤醒、关机、网络状态变化消息，则可监听以下广播消息：  
1) 待机：  
广播消息：android.intent.action.SCREEN\_OFF (代码)  
2) 唤醒：  
广播消息：android.intent.action.SCREEN\_ON (代码)  
3) 关机：  
广播消息：android.intent.action.ACTION\_SHUTDOWN (XML或代码)  
4) 网络状态变化：  
 广播消息：android.net.conn.CONNECTIVITY\_CHANGE (XML或代码)  
                  然后调用下面的isNetworkAvailable获取当前网络状态。  
public static boolean isNetworkAvailable(Context context) {   
        ConnectivityManager mgr = (ConnectivityManager) context    
                .getSystemService(Context.CONNECTIVITY\_SERVICE);    
        NetworkInfo[] info = mgr.getAllNetworkInfo();    
        if (info != null) {    
            for (int i = 0; i < info.length; i++) {    
                if (info[i].getState() == NetworkInfo.State.CONNECTED) {    
                    return true;    
                }    
            }    
        }    
        return false;    
    }

## 短按power键流程

### Android4.X标准短按power键处理流程

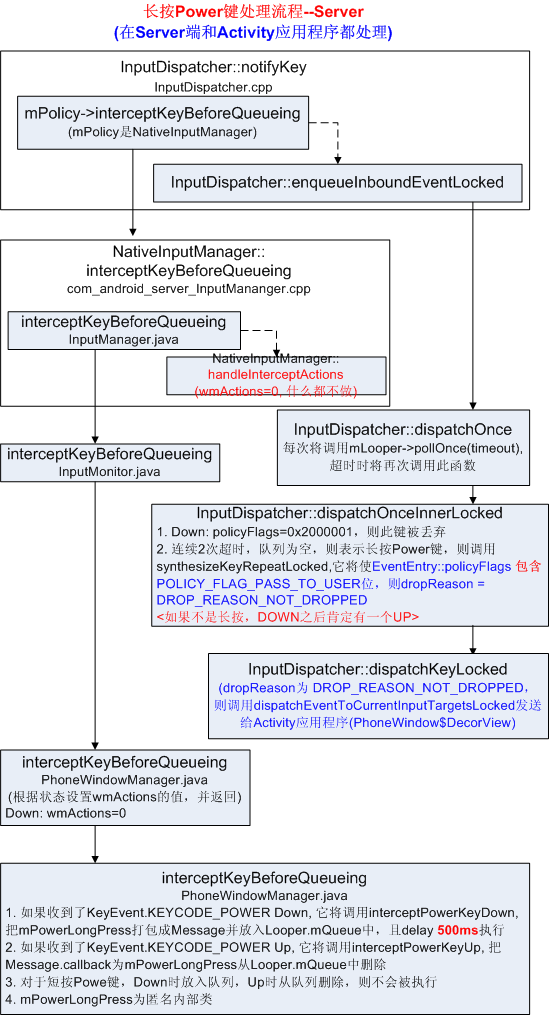


### amlogic短按power键处理流程



## 长按power键流程

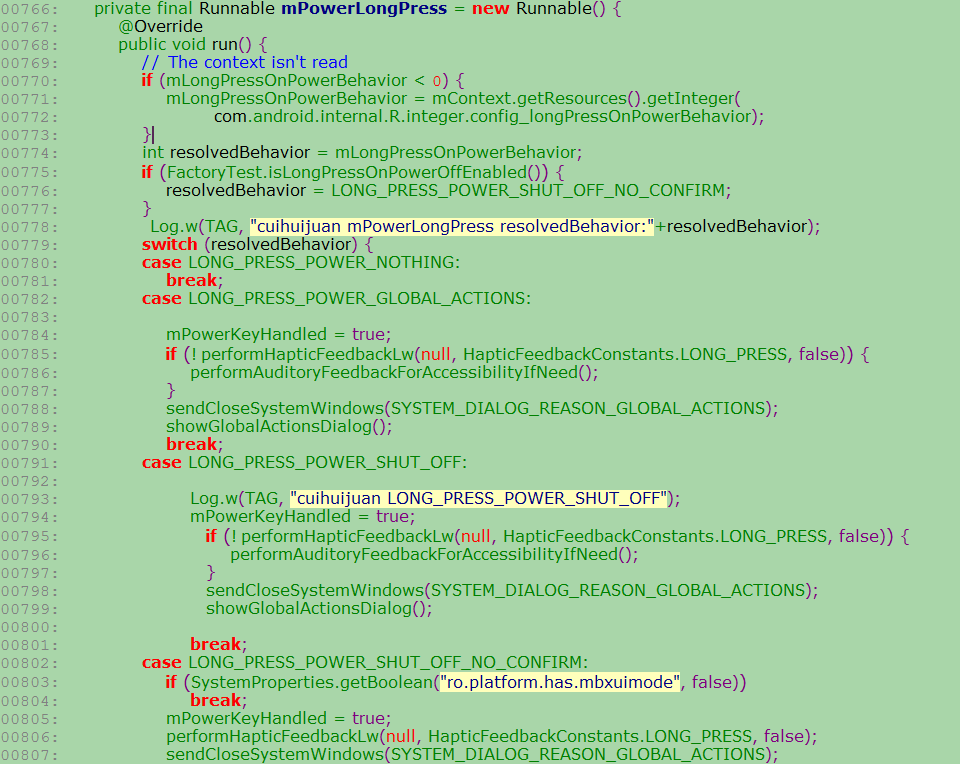
### Android4.X标准长按power键处理流程



### Amlogic长按power键处理流程

### Message超时处理流程

如果长按Power键(超过500ms)，则此消息(Message.callback为mPowerLongPress)将被执行。mPowerLongPress (PhoneWindowManager.java)定义如下：



     它是一个匿名内部类，它是一个实现Runnable的类的对象引用，因此

     new Runnable() {

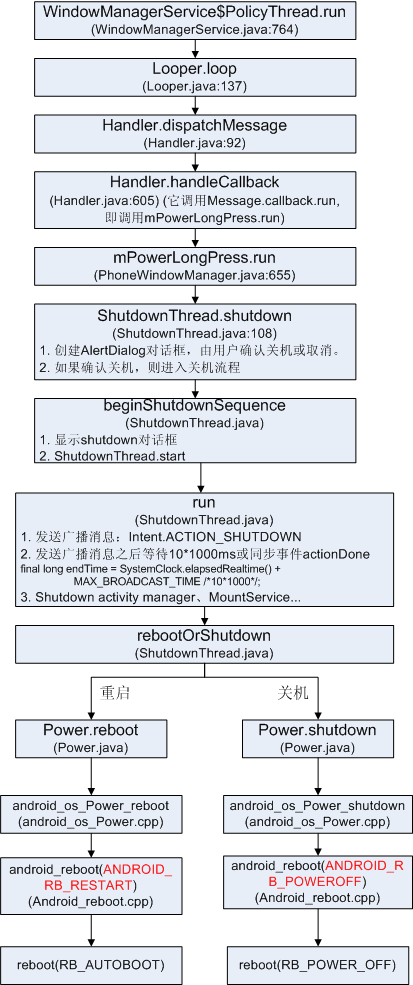
         public void run(){

         ...

         }

     };它包括了定义这个类（只不过这个类没有名字）和实例化这个类的对象。

      当超时时，其执行流程如下图所示：



# 如何处理短按和长按电源键

长按电源键：弹出关机确认对话框(KeyDown之后，如果 500ms之内，没有收到KeyUp则弹出关机确认对话框)

    短按电源键：执行待机(KeyUp时执行<wmActions=4>)或唤醒(KeyDown时执行<wmActions=2>)

对于长按电源键，在PhoneWindowManager.java的interceptKeyBeforeQueueing函数中进行处理，其相关代码如下 :

1. **case** KeyEvent.KEYCODE\_POWER: {
2. result &= ~ACTION\_PASS\_TO\_USER;
3. **if** (down) {
4. **if** (isScreenOn && !mPowerKeyTriggered
5. && (event.getFlags() & KeyEvent.FLAG\_FALLBACK) == 0) {
6. mPowerKeyTriggered = **true**;
7. mPowerKeyTime = event.getDownTime();
8. interceptScreenshotChord();
9. }
11. ...
12. // Power Key down, set mPowerLongPress executing after 500ms
13. interceptPowerKeyDown(!isScreenOn || hungUp
14. || mVolumeDownKeyTriggered || mVolumeUpKeyTriggered);
15. } **else** {
16. mPowerKeyTriggered = **false**;
17. cancelPendingScreenshotChordAction();
18. **if** (interceptPowerKeyUp(canceled || mPendingPowerKeyUpCanceled)) {
19. result = (result & ~ACTION\_POKE\_USER\_ACTIVITY) | ACTION\_GO\_TO\_SLEEP;
20. }
21. // Power key up, remove the mPowerLongPress, that is, if user release
22. // power key during 500ms, mPowerLongPress will not be execute, then execute sleep
23. mPendingPowerKeyUpCanceled = **false**;
24. }
25. **break**;
26. }

**超时处理：**

1. **private** **void** interceptPowerKeyDown(**boolean** handled) {
2. mPowerKeyHandled = handled;
3. **if** (!handled) {
4. mHandler.postDelayed(mPowerLongPress, ViewConfiguration.getGlobalActionKeyTimeout()/\*500ms\*/);
5. }
6. }
8. **private** **boolean** interceptPowerKeyUp(**boolean** canceled) {
9. **if** (!mPowerKeyHandled) {
10. mHandler.removeCallbacks(mPowerLongPress);
11. **return** !canceled;
12. }
13. **return** **false**;
14. }

# 如何短按power让上层应用自己处理

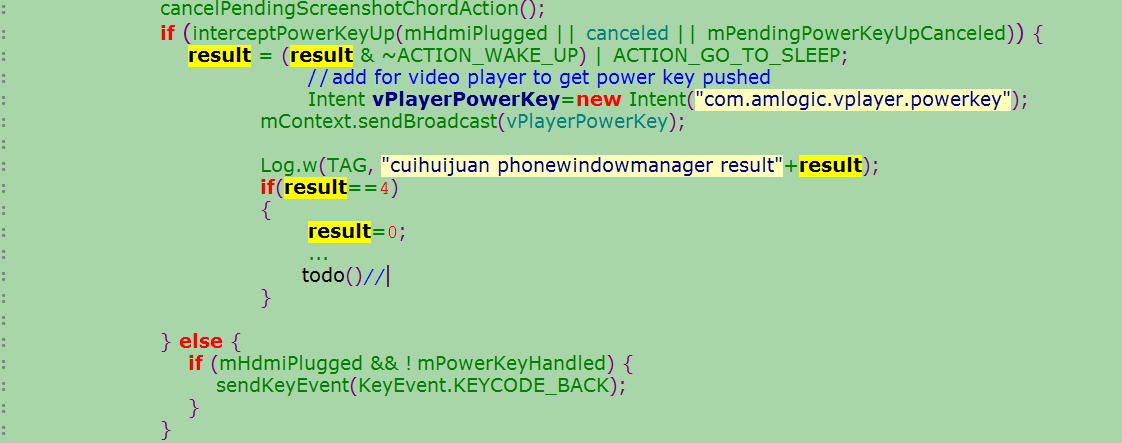
针对此问题，我们主要要解决的问题点是：

短按power键时弹出第三方具有对Power按键处理能力的应用

根据上层我们详细的短按power键的处理流程，我们可以按照下面图中的提示进行处理

所以我们只需要在PhoneWindowManager.java中的interceptKeyBeforeQueueing()接口中的

KeyEvent.KEYCODE\_POWER:处理中添加相关处理就行



在todo()那里添加我们需要检测对“com.coship.shutdown.ACTION”是否处理的action。

如果有上层应用自己去处理，如果没有我们按照android系统来处理。