Android (/tags/#Android) PowerManager (/tags/#PowerManager) Doze (/tags/#Doze)

# Android电源管理之Doze模式专题系列（十）

省电策略之屏蔽电源锁

*Posted by Cheson on April 10, 2017*

从这篇开始讲介绍Doze模式的核心作用——降低功耗的策略都是如何实现的。在状态切换剖析之Locating–>IDLE/IDLE_MAINTENANCE–>IDLE (https://chendongqi.github.io/blog/2017/04/07/pm_doze_locating_to_idle/)这一篇中已经铺段了Doze模式是如何实现降低功耗的目的，这里简单回顾下以做下文的叙述。

进入IDLE模式之后，系统会采取一系列省电策略，包括了禁止网络连接、屏蔽Wakelock、禁止同步工作、不允许Job调度和不扫描Wifi热点，这些策略就是在进入IDLE模式发出MSG_REPORT_IDLE_ON消息之后不同的服务来完成的，从本片开始就会逐个介绍，而这一篇中我们从PMS开始讲起，因为PMS为电源管理的核心服务，和Doze也有这千丝万缕的联系，就先从它开始。

线索还是得从进入IDLE模式开始讲起：当从LOCATING或者IDLE_MAINTENANCE切换到IDLE模式时，最后一个动作就是发送MSG_REPORT_IDLE_ON消息，这个消息被接受之后就会印发一些列的操作动作，这篇中我们介绍的是PMS中处理，所以只来关注第一个动作 mLocalPowerManager.setDeviceIdleMode(true)

```
case MSG_REPORT_IDLE_ON: {
    EventLogTags.writeDeviceIdleOnStart();
    mLocalPowerManager.setDeviceIdleMode(true);
    try {
        /// M: integrate Doze and App Standby @{
        if(null != getDataShapingService()) {
                mDataShapingManager.setDeviceIdleMode(true);
        }
        /// integrate Doze and App Standby @}
        mNetworkPolicyManager.setDeviceIdleMode(true);
        mBatteryStats.noteDeviceIdleMode(true, null, Process.myUid());
    } catch (RemoteException e) {
    }
    getContext().sendBroadcastAsUser(mIdleIntent, UserHandle.ALL);
    EventLogTags.writeDeviceIdleOnComplete();
} break;
```

首先需要搞清楚的是这个mLocalPowerManager和PMS的关系。它是PowerManagerInternal的对象，PowerManagerInternal.java位于framework/base/core/java/android/os/下，是一个抽象类，而在PMS中定义了一个LocalService继承自LocalPowerManager并实现了abstract方法

```
private final class LocalService extends PowerManagerInternal
```

在PowerManagerService在SystemServer中被启动时，会将本身以POWER_SERVICE这个名称注册，由ServiceManager来管理，同时也会添加一个类型为PowerManagerInternal的LocalService的服务实例

```
@Override
public void onStart() {
    publishBinderService(Context.POWER_SERVICE, new BinderService());
    publishLocalService(PowerManagerInternal.class, new LocalService());

    Watchdog.getInstance().addMonitor(this);
    Watchdog.getInstance().addThread(mHandler);
}
```

通过SystemService中接口来添加

```
/**
 * Publish the service so it is only accessible to the system process.
 */
protected final <T> void publishLocalService(Class<T> type, T service) {
    LocalServices.addService(type, service);
}
```

最后通过LocalServices中addService加入到sLocalServiceObjects这个ArrayMap中

```
/**
 * Adds a service instance of the specified interface to the global registry of local services
 */
public static <T> void addService(Class<T> type, T service) {
    synchronized (sLocalServiceObjects) {
        if (sLocalServiceObjects.containsKey(type)) {
            throw new IllegalStateException("Overriding service registration");
        }
        sLocalServiceObjects.put(type, service);
    }
}
```

如果要用这个服务的时候就如下

```
private PowerManagerInternal mLocalPowerManager;
...
mLocalPowerManager = getLocalService(PowerManagerInternal.class);
```

以上简单说明了下PowerManagerInternal这个服务的由来和注册使用方式，下面就可以随意用它了。然后言归正传，来看mLocalPowerManager.setDeviceIdleMode(true)这个调用的下文。在PMS中，这个方法实际上又直接去调用了setDeviceIdleModeInternal来处理（framework中惯用的伎俩）

```
void setDeviceIdleModeInternal(boolean enabled) {
    synchronized (mLock) {
        if (mDeviceIdleMode != enabled) {
            mDeviceIdleMode = enabled;
            updateWakeLockDisabledStatesLocked();
            if (enabled) {
                EventLogTags.writeDeviceIdleOnPhase("power");
            } else {
                EventLogTags.writeDeviceIdleOffPhase("power");
            }
        }
    }
}
```

这里把mDeviceIdleMode改成true，然后接着调用updateWakeLockDisabledStatesLocked来更新电源所的状态

```
private void updateWakeLockDisabledStatesLocked() {
    boolean changed = false;
    final int numWakeLocks = mWakeLocks.size();
    for (int i = 0; i < numWakeLocks; i++) {
        final WakeLock wakeLock = mWakeLocks.get(i);
        if ((wakeLock.mFlags & PowerManager.WAKE_LOCK_LEVEL_MASK)
                == PowerManager.PARTIAL_WAKE_LOCK) {
            if (setWakeLockDisabledStateLocked(wakeLock)) {
                changed = true;
                if (wakeLock.mDisabled) {
                    // This wake lock is no longer being respected.
                    notifyWakeLockReleasedLocked(wakeLock);
                } else {
                    notifyWakeLockAcquiredLocked(wakeLock);
                }
            }
        }
    }
    if (changed) {
        mDirty |= DIRTY_WAKE_LOCKS;
        updatePowerStateLocked();
    }
}
```

做了两件事，首先是遍历现在所有的WakeLock，针对类型为PARTIAL_WAKE_LOCK的电源锁，通过setWakeLockDisabledStateLocked方法来设置此电源锁是否可以disable

```
private boolean setWakeLockDisabledStateLocked(WakeLock wakeLock) {
    if ((wakeLock.mFlags & PowerManager.WAKE_LOCK_LEVEL_MASK)
            == PowerManager.PARTIAL_WAKE_LOCK) {
        boolean disabled = false;
        if (mDeviceIdleMode) {
            final int appid = UserHandle.getAppId(wakeLock.mOwnerUid);
            // If we are in idle mode, we will ignore all partial wake locks that are
            // for application uids that are not whitelisted.
            if (appid >= Process.FIRST_APPLICATION_UID &&
                    Arrays.binarySearch(mDeviceIdleWhitelist, appid) < 0 &&
                    Arrays.binarySearch(mDeviceIdleTempWhitelist, appid) < 0 &&
                    mUidState.get(wakeLock.mOwnerUid,                          ActivityManag
                            > ActivityManager.PROCESS_STATE_FOREGROUND_SERVICE) {
                disabled = true;
            }
        }
        if (wakeLock.mDisabled != disabled) {
            wakeLock.mDisabled = disabled;
            return true;
        }
    }
    return false;
}
```

其原理就是在mDeviceIdleMode为true时，将所有的app uid且不在白名单中程序申请的电源锁都忽略掉。设置完之后通知释放可以disable的电源锁。然后第二步就是PMS中非常常见的一个动作，更新mDirty的值，然后调用updatePowerStateLocked来更新电源状态。

PMS中做的事就如以上所介绍的，总结一下过程就是：便利所有类型为PARTIAL_WAKE_LOCK的电源锁，将其中为application uids并且不是在白名单中的程序申请的电源锁设为disable状态，然后通知释放掉disable的电源锁，最后再更新整个电源状态。

**PREVIOUS**
ANDROID电源管理之DOZE模式专题系列（八）
**(/2017/04/10/PM_DOZE_IDLE_TO_IDLEMAINTANCE/)**

**NEXT**
ANDROID电源管理之DOZE模式专题系列（九）
**(/2017/04/10/PM_DOZE_STATE_SUMMARY/)**

**FEATURED TAGS (/tags/)**

前端 (/tags/#前端) Android (/tags/#Android) frameworks (/tags/#frameworks) AlarmManager (/tags/#AlarmManager) Performance (/tags/#Performance) systrace (/tags/#systrace) PowerManager (/tags/#PowerManager) Wakelock (/tags/#Wakelock) Guitar (/tags/#Guitar) 民谣 (/tags/#民谣) 赵雷 (/tags/#赵雷) Doze (/tags/#Doze) Android Performance Patterns (/tags/#Android Performance Patterns)

**FRIENDS**

待遇见志同道合的你 (https://github.com)　小明 (http://www.betterming.cn)

● (https://twitter.com/chendongqi)

● (https://www.zhihu.com/people/chendongqi)

● (http://weibo.com/chendongqi) ● (https://www.facebook.com/chendongqi)

● (https://github.com/chendongqi)

● (https://www.linkedin.com/in/firstname-lastname-idxxxx)



Theme by Cheson (https://github.com/chendongqi/blog) | Star 1