Doze

darview

Doze

- What
- Why
- When
- Who
- How

What

What is doze?

Doze. The platform can enter a state of deep sleep (periodically resuming normal operations) if users have not actively used their device (screen off and stationary) for extended periods of time. Android 7.0 and later also enables Doze to trigger a lighter set of optimizations when users turn off the device screen yet continue to move around. (lightdeviceidle)

What does doze do?

Doze extends battery life by deferring application background CPU and network activity when a device is unused for long periods.

The platform attempts to keep the system in a sleep state, periodically resuming normal operations during a maintenance window then returning the device to sleep for longer repeating periods.

No network access, wake lock, or GPS/Wi-FI scan. Alarms and jobs deferred.

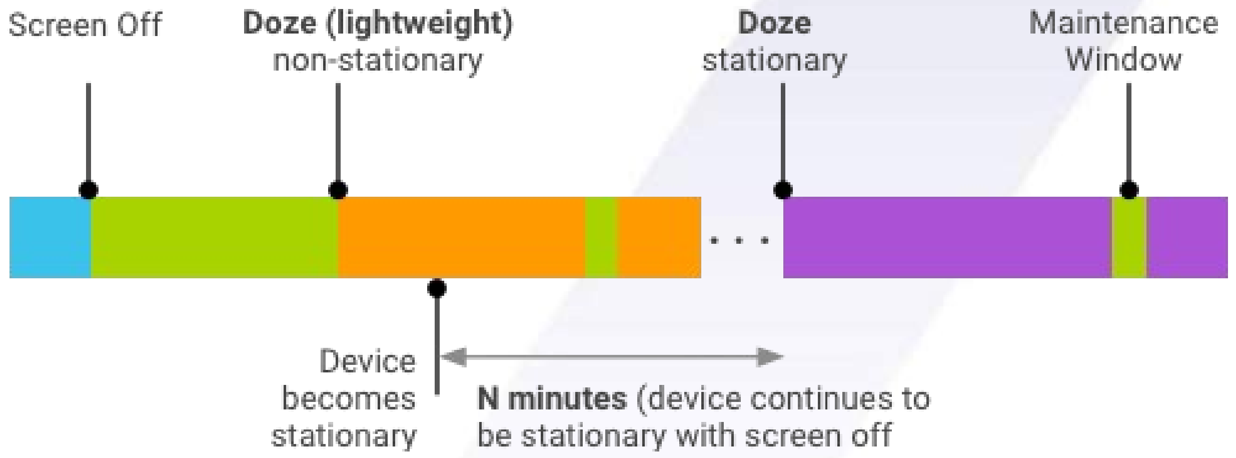
Why

Why we need doze?

Battery life is a perennial user concern.

When

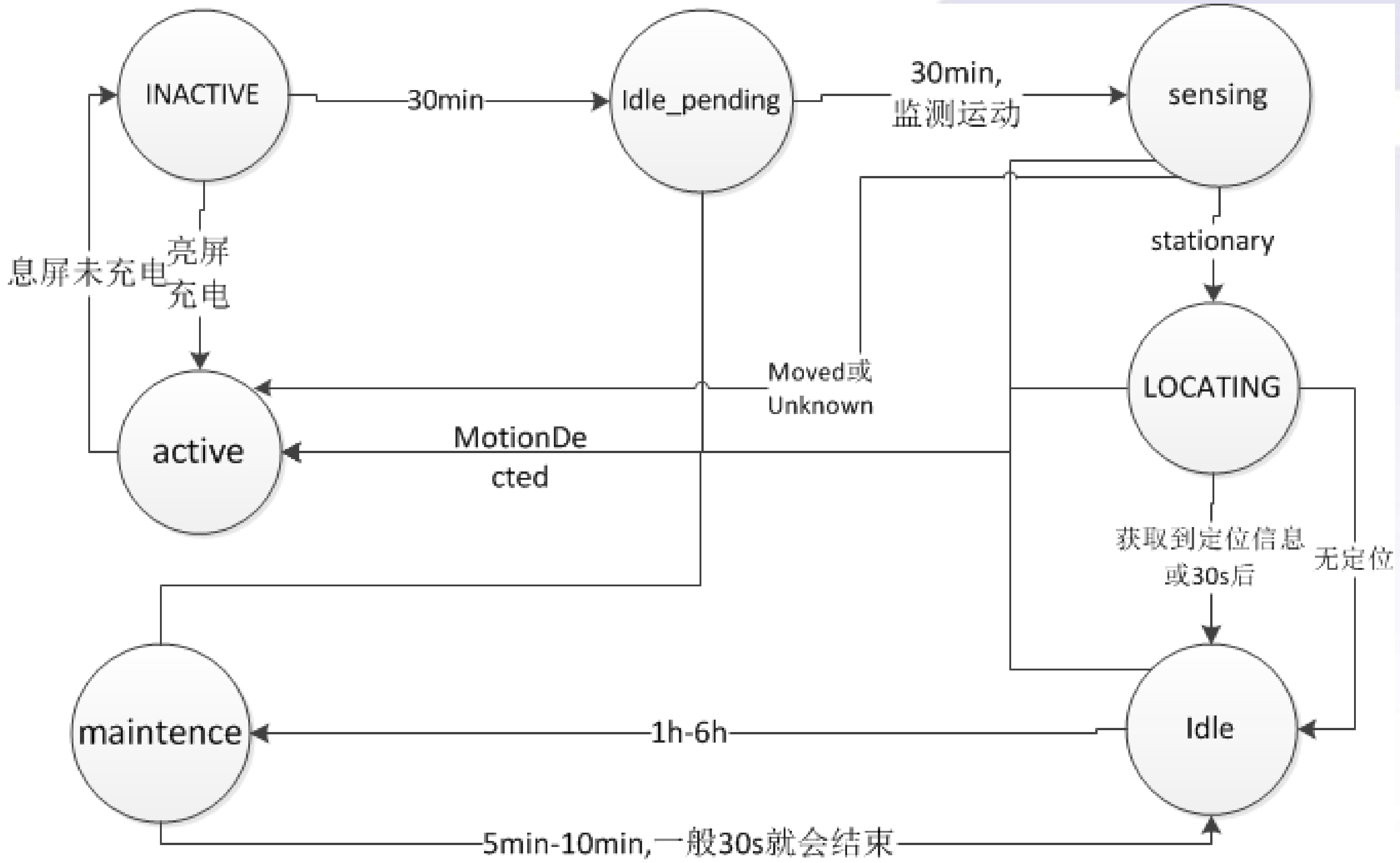
When doze start or exit?
Doze begins when the platform detects the device is idle and ends when one or more exit criteria activities occur.


Screen Off
Doze (lightweight) non-stationary
Doze stationary
Maintenance Window
Device becomes stationary
N minutes (device continues to be stationary with screen off

When-Deep_idle

Deep idle:
STATE_ACTIVE
STATE_INACTIVE
STATE_IDLE_PENDING
STATE_SENSING
STATE_LOCATING
STATE_IDLE
STATE_IDLE_MAINTENANCE

When-Deep_idle

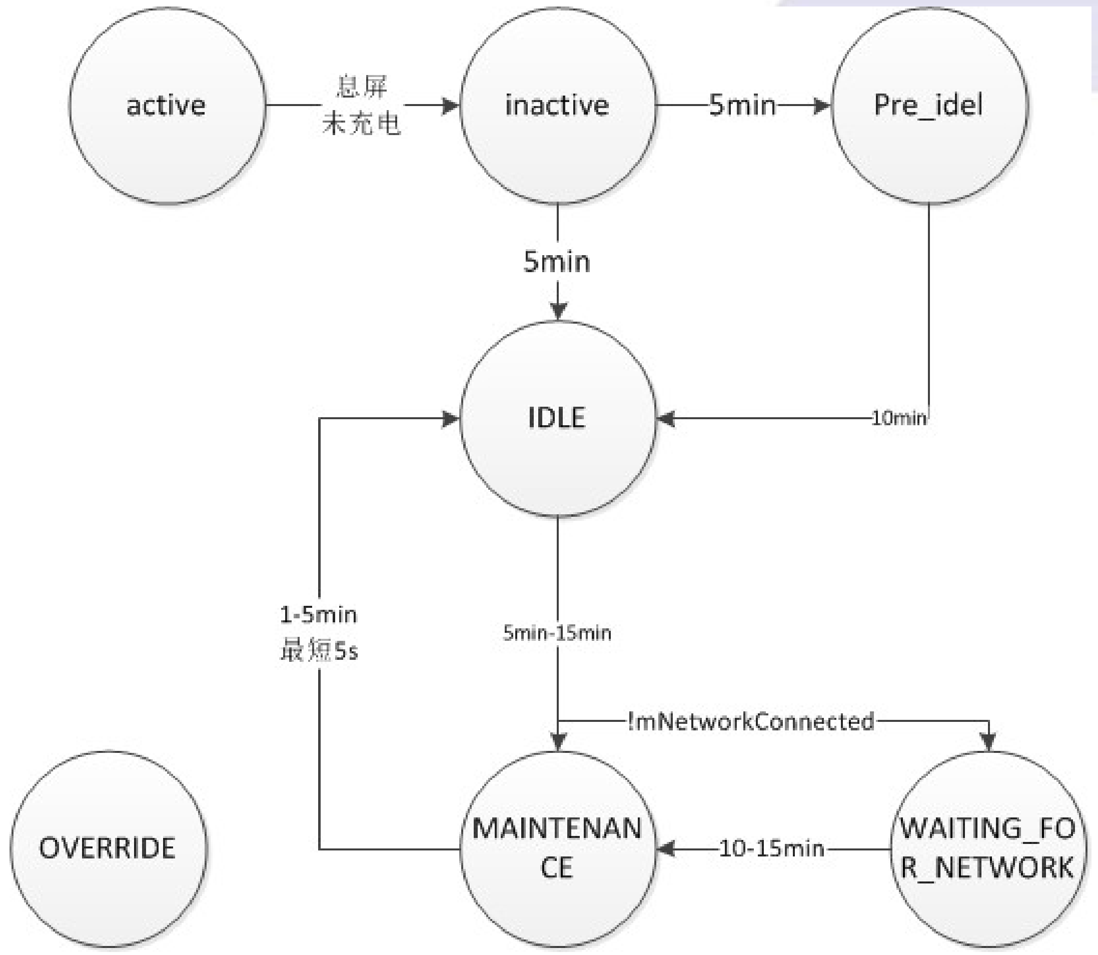


When-light_idle

lightidle:

LIGHT_STATE_ACTIVE
LIGHT_STATE_INACTIVE
LIGHT_STATE_PRE_IDLE
LIGHT_STATE_IDLE
LIGHT_STATE_WAITING_FOR_NETWORK
LIGHT_STATE_IDLE_MAINTENANCE
LIGHT_STATE_OVERRIDE

When-light_idle



Who

Who acts to doze?
Alarm:Alarms deferred(deep)
Wakelock:App wakelocks ignored(deep)

Networkpolicy:Apps not allowed network access
BatteryStats:

DeviceIdleJobsController:JobScheduler jobs deferred

UsageStatsService(deep):setAppIdleParoled(false);

GnssLocationProvider(deep):stopNavigating

WifiServiceImpl(deep):Wi-Fi scans not performed

How-alarm

How does AlarmManagerServie defer alarm?

Deviceidlecontroller 调用 AlarmManager 的 setIdleUntil 方法，使 alarmmanger 进入 doze 直到当前设置的 alarm 触发

mAlarmManager.**setIdleUnti**l(AlarmManager.ELAPSED_REALTIME_WAKEUP,
                    mNextAlarmTime, "DeviceIdleController.deep", mDeepAlarmListener,
mHandler);

这个函数最终会在 AlarmManagerService 里面将当前要设置的 alarm 设置给 mPendingIdleUntil ，
将非白名单的 alarm 加入 mPendingWhileIdleAlarms 这样一个 ArrayList 中，而不会加入到 batch 中去
当该 alarm 触发之后就会清空 mPendingIdleUntil ，或者我们删除该 alarm 的时候

正常设置的时候会将该 alarm 加入到自己对应的 batch 中去

How-Wakelock

Hot to ignore wake locks?
我们最终会调用 PowerManagerService 的 updateWakeLockDisabledStatesLocked 将
1.uid>=10000
2. 非白名单应用
3. 该 uidstate>4(ActivityManager.PROCESS_STATE_FOREGROUND_SERVICE) 的
wakeLock 的 mDisabled 位设置成 true.

从而决定是否设置 mWakeLockSummary 的 WAKE_LOCK_CPU 位，
最终决定 CPU 是否保持唤醒

如果 mWakeLockSummary 的 wake_lock_cpu 位为 1 ，就会保持 cpu alive
如果该位为 0 ，就不会持有 cpu

How-Networkpolicy

How to prevent apps netwrok access?
NetworkPolicyManager 通过 iptables 来限制网络访问

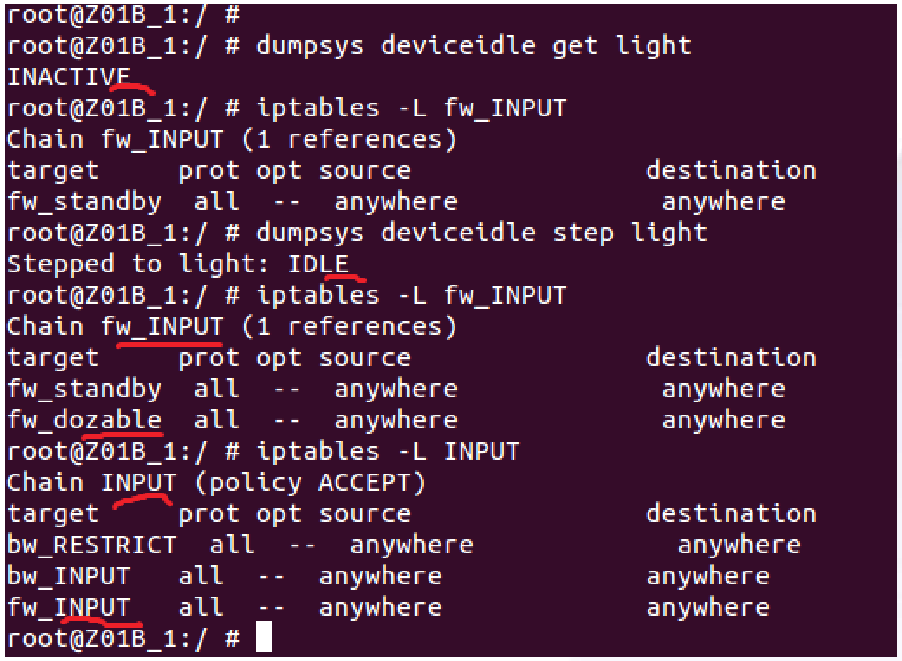
iptables 是 Linux 系统中最重要的网络管控工具。它与 Kernel 中的 netfilter 模块配合工作 , 其主要功能是为 netfilter 设置一些过滤 (filter) 或网络地址转换 (NAT) 的规则。当 Kernel 收到网络数据包后 , 将会依据 iptables 设置的规则进行相应的操作

Iptables 里面有一些 chain ， chain 里面保存着 rules
Rule 就是 iptables 工作的规则 . 首先 , 系统将检查要处理的数据包是否满足 Rule 设置的条件 , 如果满足则执行 Rule 中设置的目标 (Target), 否则继续执行 Chain 中的下一条 Rule

fw_INPUT ：接收到的数据包的处理规则的 chain
fw_OUTPUT ：要发送的数据包的处理规则的 chain
fw_dozable ： doze 模式下处理规则的 chain

我们进入 light_idle 或 deep_idle 后，会先重新设置 fw_dozable 里面的 rules ，然后将 fw_dozable 加入到 fw_input 和 fw_output 里

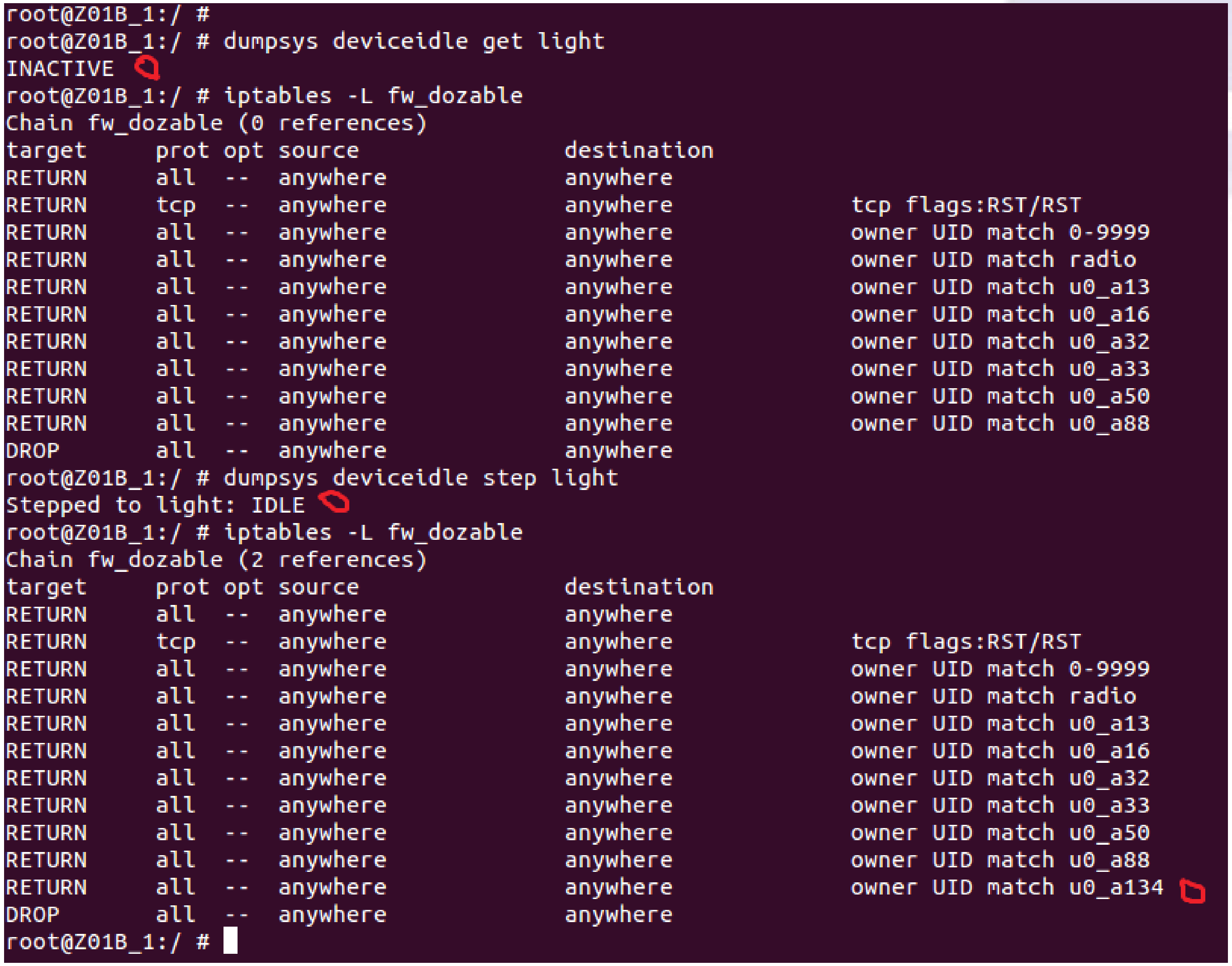
How-Networkpolicy



```
root@Z01B_1:/ #
root@Z01B_1:/ # dumpsys deviceidle get light
INACTIVE
root@Z01B_1:/ # iptables -L fw_INPUT
Chain fw_INPUT (1 references)
target     prot opt source               destination
fw_standby  all  --  anywhere             anywhere
root@Z01B_1:/ # dumpsys deviceidle step light
Stepped to light: IDLE
root@Z01B_1:/ # iptables -L fw_INPUT
Chain fw_INPUT (1 references)
target     prot opt source               destination
fw_standby  all  --  anywhere             anywhere
fw_dozable  all  --  anywhere             anywhere
root@Z01B_1:/ # iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
bw_RESTRICT  all  --  anywhere             anywhere
bw_INPUT   all  --  anywhere             anywhere
fw_INPUT   all  --  anywhere             anywhere
root@Z01B_1:/ #
```

进入 light_idle 之后，将 fw_dozable 加入了 fw_INPUT 里面，Fw_input 在 INPUT 这个 chain 里面，所以进来的数据就会根据 fw_dozable 里面的规则进行处理

How-Networkpolicy



```
root@Z01B_1:/ #
root@Z01B_1:/ # dumpsys deviceidle get light
INACTIVE
root@Z01B_1:/ # iptables -L fw_dozable
Chain fw_dozable (0 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
RETURN     tcp  --  anywhere             anywhere             tcp flags:RST/RST
RETURN     all  --  anywhere             anywhere             owner UID match 0-9999
RETURN     all  --  anywhere             anywhere             owner UID match radio
RETURN     all  --  anywhere             anywhere             owner UID match u0_a13
RETURN     all  --  anywhere             anywhere             owner UID match u0_a16
RETURN     all  --  anywhere             anywhere             owner UID match u0_a32
RETURN     all  --  anywhere             anywhere             owner UID match u0_a33
RETURN     all  --  anywhere             anywhere             owner UID match u0_a50
RETURN     all  --  anywhere             anywhere             owner UID match u0_a88
DROP       all  --  anywhere             anywhere
root@Z01B_1:/ # dumpsys deviceidle step light
Stepped to light: IDLE
root@Z01B_1:/ # iptables -L fw_dozable
Chain fw_dozable (2 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
RETURN     tcp  --  anywhere             anywhere             tcp flags:RST/RST
RETURN     all  --  anywhere             anywhere             owner UID match 0-9999
RETURN     all  --  anywhere             anywhere             owner UID match radio
RETURN     all  --  anywhere             anywhere             owner UID match u0_a13
RETURN     all  --  anywhere             anywhere             owner UID match u0_a16
RETURN     all  --  anywhere             anywhere             owner UID match u0_a32
RETURN     all  --  anywhere             anywhere             owner UID match u0_a33
RETURN     all  --  anywhere             anywhere             owner UID match u0_a50
RETURN     all  --  anywhere             anywhere             owner UID match u0_a88
RETURN     all  --  anywhere             anywhere             owner UID match u0_a134
DROP       all  --  anywhere             anywhere
root@Z01B_1:/ #
```

开启 qq 音乐之后，进入 light_idle 前后 fw_dozable 的区别

How-DeviceIdleJobsController

How JobScheduler jobs being defered?

先确保未执行的任务让它不会被执行，然后再把正在执行的工作取消

先调用 updateTaskStateLocked 更新该 job 状态，
将 satisfiedConstraints 的 CONSTRAINT_DEVICE_NOT_DOZING 位设置为 0 ；从而在 isReady 造成返回 false ，使该工作不会被执行。如果是白名单应用，则该位会被置为 1

然后再调用 onDeviceIdleStateChanged 将正在执行的工作取消掉，如果该工作 FLAG_WILL_BE_FOREGROUND 位不为 0 ，则也不会取消该工作

How

What does UsageStatsService do in deep idle mode?

setAppIdleParoled(false);
informParoleStateChanged, 将 CONSTRAINT_APP_NOT_IDLE 设置成 0
会导致 isReady 判断的时候导致返回 fasle ，从而不执行该工作

```
mAppIdleParoleOn = isAppIdleParoleOn;
GlobalUpdateFunc update = new GlobalUpdateFunc();
mJobSchedulerService.getJobStore().forEachJob(update);
```

```
String packageName = jobStatus.getSourcePackageName();
final boolean appIdle = !mAppIdleParoleOn &&
mUsageStatsInternal.isAppIdle(packageName,
        jobStatus.getSourceUid(), jobStatus.getSourceUserId());
if (jobStatus.setAppNotIdleConstraintSatisfied(!appIdle)) {
    mChanged = true;
}
```

How-GnssLocationProvider(deep)

What does GnssLocationProvider do in deep idle mode?

将 mDisableGps 置为 true ，调用 stopNavigating 函数停止 gps 定位

```
private void updateLowPowerMode() {
    // Disable GPS if we are in device idle mode.
    boolean disableGps = mPowerManager.isDeviceIdleMode();
    switch (Settings.Secure.getInt(mContext.getContentResolver(),
BATTERY_SAVER_GPS_MODE,
            BATTERY_SAVER_MODE_DISABLED_WHEN_SCREEN_OFF)) {
        case BATTERY_SAVER_MODE_DISABLED_WHEN_SCREEN_OFF:
            // If we are in battery saver mode and the screen is off, disable GPS.
            disableGps |= mPowerManager.isPowerSaveMode() && !
mPowerManager.isInteractive();
            break;
    }
    if (disableGps != mDisableGps) {
        mDisableGps = disableGps;
        updateRequirements();
    }
}
```

How-WifiServiceImpl(deep)

How wifis not not be performed?

调用 handleIdleModeChanged 将 mInIdleMode 设置成 true ，这样在下次扫描的时候会由于该位是 true 而不进行扫描

```
        if (mInIdleMode) {
            // Need to send an immediate scan result broadcast in case the
            // caller is waiting for a result ..

            // clear calling identity to send broadcast
            long callingIdentity = Binder.clearCallingIdentity();
            try {
                mWifiStateMachine.sendScanResultsAvailableBroadcast(/*
scanSucceeded = */ false);
            } finally {
                // restore calling identity
                Binder.restoreCallingIdentity(callingIdentity);
            }
            mScanPending = true;
            return;
        }
```

Significant motion

How wifis not not be performed?

A significant motion detector triggers when the detecting a “significant motion”: a motion that <u>might lead to a change in the user location.</u>

At the high level, the significant motion detector is used to reduce the power consumption of location determination. When the localization algorithms detect that the device is static, they can switch to a low power mode, where they rely on significant motion to wake the device up when the user is changing location.