PowerManager and WakeLocks

There are going to be times when you want the device to keep running, even though it ordinarily would go into a sleep mode, with the CPU powered down and the screen turned off. Sometimes, that will be based upon user interactions, or the lack thereof, such as keeping the screen on while playing back a video. Sometimes, that will be to allow background scheduled work to run to completion, as was introduced in the chapter on `AlarmManager` .

This chapter looks a bit more at the details of this sort of power management, including coverage of how `AlarmManager` works.

Prerequisites

Understanding this chapter requires that you have read the core chapters, particularly the chapter on `AlarmManager` .

Keeping the Screen On, UI-Style

If your objective is to keep the screen (and CPU) on while your activity is in the foreground, the simplest solution is to add `android:keepScreenOn="true"` to something in the activity’s layout. So long as that widget or container is visible, the screen will stay on.

If you wish to do this conditionally, `setKeepScreenOn()` allows you to toggle this setting at runtime.

Once your activity is no longer in the foreground, or the widget or container is no longer visible, the effect lapses, and screen operation returns to normal.

The Role of the WakeLock

The preview of this section was whisked away by a shark-infested tornado.

What WakefulIntentService Does

The preview of this section is unavailable right now, but if you leave your name and number at the sound of the tone, it might get back to you (BEEEEEEEEEEEEP!).