added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)
Summary: Nested Classes (#nestedclasses) | Constants (#constants) | Methods (#pubmethods) | Inherited Methods (#inhmethods) | [Expand All] (#)

PowerManager

`public final class PowerManager`
`extends Object (https://developer.android.com/reference/java/lang/Object.html)`

java.lang.Object (https://developer.android.com/reference/java/lang/Object.html)
↳ android.os.PowerManager

This class gives you control of the power state of the device.

Device battery life will be significantly affected by the use of this API. Do not acquire `PowerManager.WakeLock` `(https://developer.android.com/reference/android/os/PowerManager.WakeLock.html)`s unless you really need them, use the minimum levels possible, and be sure to release them as soon as possible.

The primary API you'll use is `newWakeLock()` `(https://developer.android.com/reference/android/os/PowerManager.html#newWakeLock(int, java.lang.String))`. This will create a `PowerManager.WakeLock` `(https://developer.android.com/reference/android/os/PowerManager.WakeLock.html)` object. You can then use methods on the wake lock object to control the power state of the device.

In practice it's quite simple:

```
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
 PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.SCREEN_DIM_WAKE_LOCK, "My Tag");
 wl.acquire();
   ..screen will stay on during this section..
 wl.release();
```

The following wake lock levels are defined, with varying effects on system power. *These levels are mutually exclusive - you may only specify one of them.*

Flag Value	CPU	Screen	Keyboard
`PARTIAL_WAKE_LOCK` `(https://developer.android.com/reference/android/os/PowerManager.html#PARTIAL_WAKE_LOCK)`	On*	Off	Off
`SCREEN_DIM_WAKE_LOCK` `(https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_DIM_WAKE_LOCK)`	On	Dim	Off
`SCREEN_BRIGHT_WAKE_LOCK` `(https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_BRIGHT_WAKE_LOCK)`	On	Bright	Off
`FULL_WAKE_LOCK` `(https://developer.android.com/reference/android/os/PowerManager.html#FULL_WAKE_LOCK)`	On	Bright	Bright

**If you hold a partial wake lock, the CPU will continue to run, regardless of any display timeouts or the state of the screen and even after the user presses the power button. In all other wake locks, the CPU will run, but the user can still put the device to sleep using the power button.*

In addition, you can add two more flags, which affect behavior of the screen only. *These flags have no effect when combined with a* `PARTIAL_WAKE_LOCK` `(https://developer.android.com/reference/android/os/PowerManager.html#PARTIAL_WAKE_LOCK)`.

Flag Value	Description
`ACQUIRE_CAUSES_WAKEUP` `(https://developer.android.com/reference/android/os/PowerManager.html#ACQUIRE_CAUSES_WAKEUP)`	Normal wake locks don't actually turn on the illumination. Instead, they cause the illumination to remain on once it turns on (e.g. from user activity). This flag will force the screen and/or keyboard to turn on immediately, when the WakeLock is acquired. A typical use would be for notifications which are important for the user to see immediately.

`ON_AFTER_RELEASE` (https://developer.android.com/reference/android/os/PowerManager.html#ON_AFTER_RELEASE)	If this flag is set, the user activity timer will be reset when the WakeLock is released, causing the illumination to remain on a bit longer. This can be used to reduce flicker if you are cycling between wake lock conditions.

Any application using a WakeLock must request the `android.permission.WAKE_LOCK` permission in an `<uses-permission>` element of the application's manifest.

Instances of this class must be obtained using `Context.getSystemService(Class)` (https://developer.android.com/reference/android/content/Context.html#getSystemService(java.lang.Class<T>)) with the argument `PowerManager.class` or `Context.getSystemService(String)` (https://developer.android.com/reference/android/content/Context.html#getSystemService(java.lang.String)) with the argument `Context.POWER_SERVICE` (https://developer.android.com/reference/android/content/Context.html#POWER_SERVICE).

Summary

Nested classes	
`class`	`PowerManager.WakeLock` (https://developer.android.com/reference/android/os/PowerManager.WakeLock.html) A wake lock is a mechanism to indicate that your application needs to have the device stay on.

Constants	
`int`	`ACQUIRE_CAUSES_WAKEUP` (https://developer.android.com/reference/android/os/PowerManager.html#ACQUIRE_CAUSES_WAKEUP) Wake lock flag: Turn the screen on when the wake lock is acquired.
`String` (https://developer.android.com/reference/java/lang/String.html)	`ACTION_DEVICE_IDLE_MODE_CHANGED` (https://developer.android.com/reference/android/os/PowerManager.html#ACTION_DEVICE_IDLE_MODE_CHANGED) Intent that is broadcast when the state of `isDeviceIdleMode()` (https://developer.android.com/reference/android/os/PowerManager.html#isDeviceIdleMode()) changes.
`String` (https://developer.android.com/reference/java/lang/String.html)	`ACTION_POWER_SAVE_MODE_CHANGED` (https://developer.android.com/reference/android/os/PowerManager.html#ACTION_POWER_SAVE_MODE_CHANGED) Intent that is broadcast when the state of `isPowerSaveMode()` (https://developer.android.com/reference/android/os/PowerManager.html#isPowerSaveMode()) changes.
`int`	`FULL_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#FULL_WAKE_LOCK) *This constant was deprecated in API level 17. Most applications should use* `FLAG_KEEP_SCREEN_ON` (https://developer.android.com/reference/android/view/WindowManager.LayoutParams.html#FLAG_KEEP_SCREEN_ON) *instead of this type of wake lock, as it will be correctly managed by the platform as the user moves between applications and doesn't require a special permission.*
`int`	`ON_AFTER_RELEASE` (https://developer.android.com/reference/android/os/PowerManager.html#ON_AFTER_RELEASE) Wake lock flag: When this wake lock is released, poke the user activity timer so the screen stays on for a little longer.
`int`	`PARTIAL_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#PARTIAL_WAKE_LOCK) Wake lock level: Ensures that the CPU is running; the screen and keyboard backlight will be allowed to go off.

int	PROXIMITY_SCREEN_OFF_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#PROXIMITY_SCREEN_OFF_WAKE_LOCK) Wake lock level: Turns the screen off when the proximity sensor activates.
int	RELEASE_FLAG_WAIT_FOR_NO_PROXIMITY (https://developer.android.com/reference/android/os/PowerManager.html#RELEASE_FLAG_WAIT_FOR_NO_PROXIMITY) Flag for WakeLock.release(int) (https://developer.android.com/reference/android/os/PowerManager.WakeLock.html#release()): Defer releasing a PROXIMITY_SCREEN_OFF_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#PROXIMITY_SCREEN_OFF_WAKE_LOCK) wake lock until the proximity sensor indicates that an object is not in close proximity.
int	SCREEN_BRIGHT_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_BRIGHT_WAKE_LOCK) *This constant was deprecated in API level 13. Most applications should use* FLAG_KEEP_SCREEN_ON (https://developer.android.com/reference/android/view/WindowManager.LayoutParams.html#FLAG_KEEP_SCREEN_ON) *instead of this type of wake lock, as it will be correctly managed by the platform as the user moves between applications and doesn't require a special permission.*
int	SCREEN_DIM_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_DIM_WAKE_LOCK) *This constant was deprecated in API level 17. Most applications should use* FLAG_KEEP_SCREEN_ON (https://developer.android.com/reference/android/view/WindowManager.LayoutParams.html#FLAG_KEEP_SCREEN_ON) *instead of this type of wake lock, as it will be correctly managed by the platform as the user moves between applications and doesn't require a special permission.*

Public methods	
boolean	isDeviceIdleMode (https://developer.android.com/reference/android/os/PowerManager.html#isDeviceIdleMode())() Returns true if the device is currently in idle mode.
boolean	isIgnoringBatteryOptimizations (https://developer.android.com/reference/android/os/PowerManager.html#isIgnoringBatteryOptimizations(java.lang.String))(String (https://developer.android.com/reference/java/lang/String.html) packageName) Return whether the given application package name is on the device's power whitelist.
boolean	isInteractive (https://developer.android.com/reference/android/os/PowerManager.html#isInteractive())() Returns true if the device is in an interactive state.
boolean	isPowerSaveMode (https://developer.android.com/reference/android/os/PowerManager.html#isPowerSaveMode())() Returns true if the device is currently in power save mode.
boolean	isScreenOn (https://developer.android.com/reference/android/os/PowerManager.html#isScreenOn())() *This method was deprecated in API level 20. Use* isInteractive() (https://developer.android.com/reference/android/os/PowerManager.html#isInteractive()) *instead.*
boolean	isSustainedPerformanceModeSupported (https://developer.android.com/reference/android/os/PowerManager.html#isSustainedPerformanceModeSupported())() This function checks if the device has implemented Sustained Performance Mode.
boolean	isWakeLockLevelSupported (https://developer.android.com/reference/android/os/PowerManager.html#isWakeLockLevelSupported(int))(int level) Returns true if the specified wake lock level is supported.

PowerManager.WakeLock (https://developer.android.com/reference/android/os/PowerManager.WakeLock.html)	newWakeLock (https://developer.android.com/reference/android/os/PowerManager.html#newWakeLock(int, java.lang.String))(int levelAndFlags, String (https://developer.android.com/reference/java/lang/String.html) tag) Creates a new wake lock with the specified level and flags.
void	reboot (https://developer.android.com/reference/android/os/PowerManager.html#reboot(java.lang.String))(String (https://developer.android.com/reference/java/lang/String.html) reason) Reboot the device.

Inherited methods
(#)From class java.lang.Object (https://developer.android.com/reference/java/lang/Object.html)

Constants

ACQUIRE_CAUSES_WAKEUP

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`int ACQUIRE_CAUSES_WAKEUP`

Wake lock flag: Turn the screen on when the wake lock is acquired.

Normally wake locks don't actually wake the device, they just cause the screen to remain on once it's already on. Think of the video player application as the normal behavior. Notifications that pop up and want the device to be on are the exception; use this flag to be like them.

Cannot be used with PARTIAL_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#PARTIAL_WAKE_LOCK).

Constant Value: 268435456 (0x10000000)

ACTION_DEVICE_IDLE_MODE_CHANGED

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`String (https://developer.android.com/reference/java/lang/String.html) ACTION_DEVICE_IDLE_MODE_CHANGED`

Intent that is broadcast when the state of isDeviceIdleMode() (https://developer.android.com/reference/android/os/PowerManager.html#isDeviceIdleMode()) changes. This broadcast is only sent to registered receivers.

Constant Value: "android.os.action.DEVICE_IDLE_MODE_CHANGED"

ACTION_POWER_SAVE_MODE_CHANGED

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`String (https://developer.android.com/reference/java/lang/String.html) ACTION_POWER_SAVE_MODE_CHANGED`

Intent that is broadcast when the state of isPowerSaveMode() (https://developer.android.com/reference/android/os/PowerManager.html#isPowerSaveMode()) changes. This broadcast is only sent to registered receivers.

Constant Value: "android.os.action.POWER_SAVE_MODE_CHANGED"

FULL_WAKE_LOCK

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`int FULL_WAKE_LOCK`

> **This constant was deprecated in API level 17.**

> Most applications should use FLAG_KEEP_SCREEN_ON (https://developer.android.com/reference/android/view/WindowManager.LayoutParams.html#FLAG_KEEP_SCREEN_ON) instead of this type of wake lock, as it will be correctly managed by the platform as the user moves between applications and doesn't require a special permission.

Wake lock level: Ensures that the screen and keyboard backlight are on at full brightness.

If the user presses the power button, then the FULL_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#FULL_WAKE_LOCK) will be implicitly released by the system, causing both the screen and the CPU to be turned off. Contrast with PARTIAL_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#PARTIAL_WAKE_LOCK).

Constant Value: 26 (0x0000001a)

ON_AFTER_RELEASE

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
int ON_AFTER_RELEASE
```

Wake lock flag: When this wake lock is released, poke the user activity timer so the screen stays on for a little longer.

Will not turn the screen on if it is not already on. See ACQUIRE_CAUSES_WAKEUP (https://developer.android.com/reference/android/os/PowerManager.html#ACQUIRE_CAUSES_WAKEUP) if you want that.

Cannot be used with PARTIAL_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#PARTIAL_WAKE_LOCK).

Constant Value: 536870912 (0x20000000)

PARTIAL_WAKE_LOCK

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
int PARTIAL_WAKE_LOCK
```

Wake lock level: Ensures that the CPU is running; the screen and keyboard backlight will be allowed to go off.

If the user presses the power button, then the screen will be turned off but the CPU will be kept on until all partial wake locks have been released.

Constant Value: 1 (0x00000001)

PROXIMITY_SCREEN_OFF_WAKE_LOCK

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
int PROXIMITY_SCREEN_OFF_WAKE_LOCK
```

Wake lock level: Turns the screen off when the proximity sensor activates.

If the proximity sensor detects that an object is nearby, the screen turns off immediately. Shortly after the object moves away, the screen turns on again.

A proximity wake lock does not prevent the device from falling asleep unlike FULL_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#FULL_WAKE_LOCK), SCREEN_BRIGHT_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_BRIGHT_WAKE_LOCK) and SCREEN_DIM_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_DIM_WAKE_LOCK). If there is no user activity and no other wake locks are held, then the device will fall asleep (and lock) as usual. However, the device will not fall asleep while the screen has been turned off by the proximity sensor because it effectively counts as ongoing user activity.

Since not all devices have proximity sensors, use isWakeLockLevelSupported(int) (https://developer.android.com/reference/android/os/PowerManager.html#isWakeLockLevelSupported(int)) to determine whether this wake lock level is supported.

Cannot be used with ACQUIRE_CAUSES_WAKEUP (https://developer.android.com/reference/android/os/PowerManager.html#ACQUIRE_CAUSES_WAKEUP).

Constant Value: 32 (0x00000020)

RELEASE_FLAG_WAIT_FOR_NO_PROXIMITY

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`int RELEASE_FLAG_WAIT_FOR_NO_PROXIMITY`

Flag for `WakeLock.release(int)` (https://developer.android.com/reference/android/os/PowerManager.WakeLock.html#release()): Defer releasing a `PROXIMITY_SCREEN_OFF_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#PROXIMITY_SCREEN_OFF_WAKE_LOCK) wake lock until the proximity sensor indicates that an object is not in close proximity.

Constant Value: 1 (0x00000001)

SCREEN_BRIGHT_WAKE_LOCK

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`int SCREEN_BRIGHT_WAKE_LOCK`

> **This constant was deprecated in API level 13.**
> Most applications should use `FLAG_KEEP_SCREEN_ON` (https://developer.android.com/reference/android/view/WindowManager.LayoutParams.html#FLAG_KEEP_SCREEN_ON) instead of this type of wake lock, as it will be correctly managed by the platform as the user moves between applications and doesn't require a special permission.

Wake lock level: Ensures that the screen is on at full brightness; the keyboard backlight will be allowed to go off.

If the user presses the power button, then the `SCREEN_BRIGHT_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_BRIGHT_WAKE_LOCK) will be implicitly released by the system, causing both the screen and the CPU to be turned off. Contrast with `PARTIAL_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#PARTIAL_WAKE_LOCK).

Constant Value: 10 (0x0000000a)

SCREEN_DIM_WAKE_LOCK

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`int SCREEN_DIM_WAKE_LOCK`

> **This constant was deprecated in API level 17.**
> Most applications should use `FLAG_KEEP_SCREEN_ON` (https://developer.android.com/reference/android/view/WindowManager.LayoutParams.html#FLAG_KEEP_SCREEN_ON) instead of this type of wake lock, as it will be correctly managed by the platform as the user moves between applications and doesn't require a special permission.

Wake lock level: Ensures that the screen is on (but may be dimmed); the keyboard backlight will be allowed to go off.

If the user presses the power button, then the `SCREEN_DIM_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_DIM_WAKE_LOCK) will be implicitly released by the system, causing both the screen and the CPU to be turned off. Contrast with `PARTIAL_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#PARTIAL_WAKE_LOCK).

Constant Value: 6 (0x00000006)

Public methods

isDeviceIdleMode

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`boolean isDeviceIdleMode ()`

Returns true if the device is currently in idle mode. This happens when a device has been sitting unused and unmoving for a sufficiently long period of time, so that it decides to go into a lower power-use state. This may involve things like turning off network access to apps. You can monitor for changes to this state with `ACTION_DEVICE_IDLE_MODE_CHANGED` (https://developer.android.com/reference/android/os/PowerManager.html#ACTION_DEVICE_IDLE_MODE_CHANGED).

Returns	
`boolean`	Returns true if currently in active device idle mode, else false. This is when idle mode restrictions are being actively applied; it will return false if the device is in a long-term idle mode but currently running a maintenance window where restrictions have been lifted.

isIgnoringBatteryOptimizations

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
boolean isIgnoringBatteryOptimizations (String (https://developer.android.com/reference/java/lang/String
```

Return whether the given application package name is on the device's power whitelist. Apps can be placed on the whitelist through the settings UI invoked by `ACTION_IGNORE_BATTERY_OPTIMIZATION_SETTINGS` (https://developer.android.com/reference/android/provider/Settings.html#ACTION_IGNORE_BATTERY_OPTIMIZATION_SETTINGS).

Parameters	
`packageName`	`String`

Returns	
`boolean`	

isInteractive

added in API level 20 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
boolean isInteractive ()
```

Returns true if the device is in an interactive state.

When this method returns true, the device is awake and ready to interact with the user (although this is not a guarantee that the user is actively interacting with the device just this moment). The main screen is usually turned on while in this state. Certain features, such as the proximity sensor, may temporarily turn off the screen while still leaving the device in an interactive state. Note in particular that the device is still considered to be interactive while dreaming (since dreams can be interactive) but not when it is dozing or asleep.

When this method returns false, the device is dozing or asleep and must be awoken before it will become ready to interact with the user again. The main screen is usually turned off while in this state. Certain features, such as "ambient mode" may cause the main screen to remain on (albeit in a low power state) to display system-provided content while the device dozes.

The system will send a `screen on` (https://developer.android.com/reference/android/content/Intent.html#ACTION_SCREEN_ON) or `screen off` (https://developer.android.com/reference/android/content/Intent.html#ACTION_SCREEN_OFF) broadcast whenever the interactive state of the device changes. For historical reasons, the names of these broadcasts refer to the power state of the screen but they are actually sent in response to changes in the overall interactive state of the device, as described by this method.

Services may use the non-interactive state as a hint to conserve power since the user is not present.

Returns	
`boolean`	True if the device is in an interactive state.

See also:

`ACTION_SCREEN_ON` (https://developer.android.com/reference/android/content/Intent.html#ACTION_SCREEN_ON)

`ACTION_SCREEN_OFF` (https://developer.android.com/reference/android/content/Intent.html#ACTION_SCREEN_OFF)

isPowerSaveMode

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`boolean isPowerSaveMode ()`

Returns true if the device is currently in power save mode. When in this mode, applications should reduce their functionality in order to conserve battery as much as possible. You can monitor for changes to this state with `ACTION_POWER_SAVE_MODE_CHANGED` (https://developer.android.com/reference/android/os/PowerManager.html#ACTION_POWER_SAVE_MODE_CHANGED).

Returns	
`boolean`	Returns true if currently in low power mode, else false.

isScreenOn

added in API level 7 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`boolean isScreenOn ()`

> **This method was deprecated in API level 20.**
> Use `isInteractive()` (https://developer.android.com/reference/android/os/PowerManager.html#isInteractive()) instead.

Returns true if the device is in an interactive state.

For historical reasons, the name of this method refers to the power state of the screen but it actually describes the overall interactive state of the device. This method has been replaced by `isInteractive()` (https://developer.android.com/reference/android/os/PowerManager.html#isInteractive()).

The value returned by this method only indicates whether the device is in an interactive state which may have nothing to do with the screen being on or off. To determine the actual state of the screen, use `getState()` (https://developer.android.com/reference/android/view/Display.html#getState()).

Returns	
`boolean`	True if the device is in an interactive state.

isSustainedPerformanceModeSupported

added in API level 24 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`boolean isSustainedPerformanceModeSupported ()`

This function checks if the device has implemented Sustained Performance Mode. This needs to be checked only once and is constant for a particular device/release. Sustained Performance Mode is intended to provide a consistent level of performance for prolonged amount of time. Applications should check if the device supports this mode, before using `setSustainedPerformanceMode(boolean)` (https://developer.android.com/reference/android/view/Window.html#setSustainedPerformanceMode(boolean)).

Returns	
`boolean`	Returns True if the device supports it, false otherwise.

See also:

`setSustainedPerformanceMode(boolean)` (https://developer.android.com/reference/android/view/Window.html#setSustainedPerformanceMode(boolean))

isWakeLockLevelSupported

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`boolean isWakeLockLevelSupported (int level)`

Returns true if the specified wake lock level is supported.

Parameters

`level`	`int`: The wake lock level to check.

Returns	
`boolean`	True if the specified wake lock level is supported.

newWakeLock

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
PowerManager.WakeLock (https://developer.android.com/reference/android/os/PowerManager.WakeLock.html) newWake
                String (https://developer.android.com/reference/java/lang/String.html) tag)
```

Creates a new wake lock with the specified level and flags.

The `levelAndFlags` parameter specifies a wake lock level and optional flags combined using the logical OR operator.

The wake lock levels are: `PARTIAL_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#PARTIAL_WAKE_LOCK), `FULL_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#FULL_WAKE_LOCK), `SCREEN_DIM_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_DIM_WAKE_LOCK) and `SCREEN_BRIGHT_WAKE_LOCK` (https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_BRIGHT_WAKE_LOCK). Exactly one wake lock level must be specified as part of the `levelAndFlags` parameter.

The wake lock flags are: `ACQUIRE_CAUSES_WAKEUP` (https://developer.android.com/reference/android/os/PowerManager.html#ACQUIRE_CAUSES_WAKEUP) and `ON_AFTER_RELEASE` (https://developer.android.com/reference/android/os/PowerManager.html#ON_AFTER_RELEASE). Multiple flags can be combined as part of the `levelAndFlags` parameters.

Call `acquire()` (https://developer.android.com/reference/android/os/PowerManager.WakeLock.html#acquire()) on the object to acquire the wake lock, and `release()` (https://developer.android.com/reference/android/os/PowerManager.WakeLock.html#release()) when you are done.

```
PowerManager pm = (PowerManager)mContext.getSystemService(
                                          Context.POWER_SERVICE);
 PowerManager.WakeLock wl = pm.newWakeLock(
                                      PowerManager.SCREEN_DIM_WAKE_LOCK
                                      | PowerManager.ON_AFTER_RELEASE,
                                      TAG);
 wl.acquire();
 // ... do work...
 wl.release();
```

Although a wake lock can be created without special permissions, the `WAKE_LOCK` (https://developer.android.com/reference/android/Manifest.permission.html#WAKE_LOCK) permission is required to actually acquire or release the wake lock that is returned.

> If using this to keep the screen on, you should strongly consider using `FLAG_KEEP_SCREEN_ON` (https://developer.android.com/reference/android/view/WindowManager.LayoutParams.html#FLAG_KEEP_SCREEN_ON) instead. This window flag will be correctly managed by the platform as the user moves between applications and doesn't require a special permission.

Parameters	
`levelAndFlags`	`int`: Combination of wake lock level and flag values defining the requested behavior of the WakeLock.
`tag`	`String`: Your class name (or other tag) for debugging purposes.

Returns	
`PowerManager.WakeLock` (https://developer.android.com/reference/android/os/PowerManager.WakeLock.html)	

See also:

acquire() (https://developer.android.com/reference/android/os/PowerManager.WakeLock.html#acquire())

release() (https://developer.android.com/reference/android/os/PowerManager.WakeLock.html#release())

PARTIAL_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#PARTIAL_WAKE_LOCK)

FULL_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#FULL_WAKE_LOCK)

SCREEN_DIM_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_DIM_WAKE_LOCK)

SCREEN_BRIGHT_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#SCREEN_BRIGHT_WAKE_LOCK)

PROXIMITY_SCREEN_OFF_WAKE_LOCK (https://developer.android.com/reference/android/os/PowerManager.html#PROXIMITY_SCREEN_OFF_WAKE_LOCK)

ACQUIRE_CAUSES_WAKEUP (https://developer.android.com/reference/android/os/PowerManager.html#ACQUIRE_CAUSES_WAKEUP)

ON_AFTER_RELEASE (https://developer.android.com/reference/android/os/PowerManager.html#ON_AFTER_RELEASE)

reboot

added in API level 8 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void reboot (String (https://developer.android.com/reference/java/lang/String.html) reason)
```

Reboot the device. Will not return if the reboot is successful.

Requires the REBOOT (https://developer.android.com/reference/android/Manifest.permission.html#REBOOT) permission.

Parameters	
reason	String: code to pass to the kernel (e.g., "recovery") to request special boot modes, or null.