Technical Q&A QA1340

Registering and unregistering for sleep and wake notifications

Q: How can my application get notified when the computer is going to sleep or waking from sleep? How to I prevent sleep?

A: Cocoa (Listing 1) and I/O Kit (Listing 3, Listing 4) can both be used to receive sleep and wake notifications. Cocoa can be used to receive sleep and wake notifications, while I/O Kit can also prevent or delay (Listing 2, Listing 3) idle sleep. However, even with I/O Kit, it is not possible to prevent forced sleep, only delay it.

> **Note:** Mac OS X will sleep in two different situations- forced and idle.

- Forced sleep occurs when the user takes some sort of direct action to cause the machine to sleep. Closing the lid on a laptop or selecting sleep from the Apple menu both cause forced sleep. The system will also induce forced sleep under certain conditions, for example, a thermal emergency or a low battery.
- Idle sleep occurs when the machine is unused for a specific period of time configured in the Energy Saver System Preferences.

Listing 1: Installing a Cocoa sleep and wake notification.

```
- (void) receiveSleepNote: (NSNotification*) note
{
    NSLog(@"receiveSleepNote: %@", [note name]);
}

- (void) receiveWakeNote: (NSNotification*) note
{
    NSLog(@"receiveWakeNote: %@", [note name]);
}

- (void) fileNotifications
{
    //These notifications are filed on NSWorkspace's notification center, not the
default
    // notification center. You will not receive sleep/wake notifications if you
file
    //with the default notification center.
    [[[NSWorkspace sharedWorkspace] notificationCenter] addObserver: self
            selector: @selector(receiveSleepNote:)
            name: NSWorkspaceWillSleepNotification object: NULL];
```

```
    [[[NSWorkspace sharedWorkspace] notificationCenter] addObserver: self
            selector: @selector(receiveWakeNote:)
            name: NSWorkspaceDidWakeNotification object: NULL];
}
```

Note: `IOPMAssertionCreateWithName` is new API available in Mac OS X 10.6 Snow Leopard. `IOPMAssertionCreateWithName` allows an application to return a return a brief string to the user explaining why that application is preventing sleep. If you need to support previous versions of Mac OS X you will need to use the notification based APIs (Listing 3, Listing 4) or the deprecated call `IOPMAssertionCreate`.

Listing 2: Preventing sleep using I/O Kit in Mac OS X 10.6 Snow Leopard

```
...

#import <IOKit/pwr_mgt/IOPMLib.h>

...
// kIOPMAssertionTypeNoDisplaySleep prevents display sleep,
// kIOPMAssertionTypeNoIdleSleep prevents idle sleep

//reasonForActivity is a descriptive string used by the system whenever it needs
//  to tell the user why the system is not sleeping. For example,
//  "Mail Compacting Mailboxes" would be a useful string.

//  NOTE: IOPMAssertionCreateWithName limits the string to 128 characters.
CFStringRef* reasonForActivity= CFSTR("Describe Activity Type");

IOPMAssertionID assertionID;
IOReturn success = IOPMAssertionCreateWithName(kIOPMAssertionTypeNoDisplaySleep,
                                    kIOPMAssertionLevelOn, reasonForActivity,
&assertionID);
if (success == kIOReturnSuccess)
{

    //Add the work you need to do without
    //  the system sleeping here.

    success = IOPMAssertionRelease(assertionID);
    //The system will be able to sleep again.
}
...
```

Listing 3: Installing an I/O Kit sleep/wake notification.

```
#include <ctype.h>
#include <stdlib.h>
#include <stdio.h>
```

```
#include <mach/mach_port.h>
#include <mach/mach_interface.h>
#include <mach/mach_init.h>

#include <IOKit/pwr_mgt/IOPMLib.h>
#include <IOKit/IOMessage.h>

io_connect_t  root_port; // a reference to the Root Power Domain IOService


void
MySleepCallBack( void * refCon, io_service_t service, natural_t messageType, void *
messageArgument )
{
    printf( "messageType %08lx, arg %08lx\n",
        (long unsigned int)messageType,
        (long unsigned int)messageArgument );

    switch ( messageType )
    {

        case kIOMessageCanSystemSleep:
            /* Idle sleep is about to kick in. This message will not be sent for
forced sleep.
                Applications have a chance to prevent sleep by calling
IOCancelPowerChange.
                Most applications should not prevent idle sleep.

                Power Management waits up to 30 seconds for you to either allow or
deny idle
                sleep. If you don't acknowledge this power change by calling either
                IOAllowPowerChange or IOCancelPowerChange, the system will wait 30
                seconds then go to sleep.
            */

            //Uncomment to cancel idle sleep
            //IOCancelPowerChange( root_port, (long)messageArgument );
            // we will allow idle sleep
            IOAllowPowerChange( root_port, (long)messageArgument );
            break;

        case kIOMessageSystemWillSleep:
            /* The system WILL go to sleep. If you do not call IOAllowPowerChange or
                IOCancelPowerChange to acknowledge this message, sleep will be
                delayed by 30 seconds.

                NOTE: If you call IOCancelPowerChange to deny sleep it returns
                kIOReturnSuccess, however the system WILL still go to sleep.
            */
```

```
            IOAllowPowerChange( root_port, (long)messageArgument );
            break;

        case kIOMessageSystemWillPowerOn:
            //System has started the wake up process...
            break;

        case kIOMessageSystemHasPoweredOn:
            //System has finished waking up...
        break;

        default:
            break;

    }
}



int main( int argc, char **argv )
{
    // notification port allocated by IORegisterForSystemPower
    IONotificationPortRef  notifyPortRef;

    // notifier object, used to deregister later
    io_object_t            notifierObject;
   // this parameter is passed to the callback
    void*                  refCon;

    // register to receive system sleep notifications

    root_port = IORegisterForSystemPower( refCon, &notifyPortRef, MySleepCallBack,
&notifierObject );
    if ( root_port == 0 )
    {
        printf("IORegisterForSystemPower failed\n");
        return 1;
    }

    // add the notification port to the application runloop
    CFRunLoopAddSource( CFRunLoopGetCurrent(),
            IONotificationPortGetRunLoopSource(notifyPortRef), kCFRunLoopCommonModes
);

    /* Start the run loop to receive sleep notifications. Don't call CFRunLoopRun if
this code
        is running on the main thread of a Cocoa or Carbon application. Cocoa and
Carbon
        manage the main thread's run loop for you as part of their event handling
```

```
        mechanisms.
    */
    CFRunLoopRun();

    //Not reached, CFRunLoopRun doesn't return in this case.
    return (0);
}
```

To stop receiving I/O Kit sleep notifications, you need to remove your event source from the application runloop and do a bit of cleanup.

Listing 4: Removing I/O Kit sleep/wake notification handler.

```
...
    // we no longer want sleep notifications:

    // remove the sleep notification port from the application runloop
    CFRunLoopRemoveSource( CFRunLoopGetCurrent(),
                           IONotificationPortGetRunLoopSource(notifyPortRef),
                           kCFRunLoopCommonModes );

    // deregister for system sleep notifications
    IODeregisterForSystemPower( &notifierObject );

    // IORegisterForSystemPower implicitly opens the Root Power Domain IOService
    // so we close it here
    IOServiceClose( root_port );

    // destroy the notification port allocated by IORegisterForSystemPower
    IONotificationPortDestroy( notifyPortRef );
...
```

Document Revision History

Date	Notes
2014-01-13	Corrected small bug in message logged in receiveWakeNote
2011-03-01	Updated code to use IOPMAssertionCreateWithName.
2008-08-08	Added information about forced/idle sleep, how to receive sleep/wake notification in Cocoa, and expanded IOKit sample to include wake messages and IOPMAssert.
2005-10-17	Added an example of how to deregister for sleep notifications.
2004-10-25	New document that explains how applications can register and unregister for sleep and wake notifications on Mac OS X.