The Science of Sleep

Ensuring Application Compatibility with System Sleep and Display Power Management

June 16, 2010

Abstract

Windows® power management is becoming an increasingly important component of Windows. Users and informational technology (IT) professionals rely on features such as automatic system sleep and display power management to save energy and extend the battery life of laptops and notebooks. Developers of Windows applications, services, and drivers must ensure that their applications honor and work intelligently with power management–related user and IT administrator preferences and policies. Additionally, developers must ensure that their software works appropriately through frequent sleep/resume transitions.

Developers whose applications do not work well with Windows power management risk having their applications excluded from PCs that are owned by both individuals and organizations. This paper covers high-level best practices that software developers should follow to ensure that their application, service, or driver is compatible with and takes advantage of sleep and display power management in Windows.

This information applies to the Windows 7 operating system. Although the APIs and tools are also available in Windows Server® 2008 R2, most server-class hardware does not currently support or is not configured for automatic sleep. Many APIs and tools are also supported on earlier versions of Windows and Windows Server.

References and resources discussed here are listed at the end of this paper.

The current version of this paper is maintained on the Web at:   
 <http://www.microsoft.com/whdc/system/pnppwr/powermgmt/Science-Sleep.mspx>

**Disclaimer**: This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2010 Microsoft Corporation. All rights reserved.

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| June 16, 2010 | First publication | | | |

Contents

[Introduction 3](#_Toc264470100)

[Availability Requests 3](#_Toc264470101)

[Best Practices for Availability Requests 4](#_Toc264470102)

[Best Practices for Scenarios that Involve Passive Viewing of Content 5](#_Toc264470103)

[Best Practices for Long-Running Processes 5](#_Toc264470104)

[Using Availability Requests When on Battery Power 6](#_Toc264470105)

[“Energy Smart” Application Summary 6](#_Toc264470106)

[Availability Requests Diagnostic Tools 6](#_Toc264470107)

[Supporting Sleep Transitions 7](#_Toc264470108)

[Suspend Phase 7](#_Toc264470109)

[Resume Phase 8](#_Toc264470110)

[Sleep/Hibernate Transition Recommendations for Application Developers 9](#_Toc264470111)

[Sleep/Hibernate Transition Recommendations for Service Developers 10](#_Toc264470112)

[Sleep/Hibernate Transition Recommendations for Driver Developers 11](#_Toc264470113)

[Sleep/Hibernate Transition Recommendation for PC System Manufacturers 12](#_Toc264470114)

[Summary 12](#_Toc264470115)

[Resources 13](#_Toc264470116)

# Introduction

By default, Windows®-based platforms enable device and system power management technologies that help improve power efficiency and reduce the overall energy use of PCs. Two of the most effective power management features are *display power management* and *automatic sleep*, both of which can dramatically reduce the power use of an idle system.

Display power management—powering off or reducing brightness of the display after a period of user inactivity—is effective because the display consumes a large amount of the total system power budget. Similarly, automatic sleep after user inactivity is an effective power-saving feature because it reduces the power consumption of a typical idle PC to only a few watts.

Mobile users have long relied on their computer’s ability to quickly perform sleep transitions. However, both individuals and organizations increasingly demand that all PCs, including desktops, can enter a sleep state. In addition to manual sleep/resume transitions, Windows provides the ability for PCs to automatically enter a sleep state if no user activity is detected or to remotely be placed in sleep and awakened.

Some PCs have aggressive sleep time-outs to save power. To provide a positive user experience on these machines, applications, services, and drivers must be able to perform reliably through sleep transition events and not unnecessarily consume resources immediately after the PC resumes from sleep.

In some scenarios, such as passive viewing and lengthy data transfers, applications or drivers should temporarily disable power management technologies so that they perform tasks as users expect. Applications can temporarily disable display power management and automatic sleep by making availability requests.

However, the unwanted side-effect of availability requests is that applications or drivers can cause the machine or display to stay awake unnecessarily, which wastes power and drains batteries. This can be caused not only by the direct misuse of availability requests, but also by the use of another component that creates an availability request that the application developer is not aware of.

Developers, testers, and IT professionals should take advantage of the built-in diagnostic tools that Windows 7 offers to help determine what prevents displays from dimming and systems from automatically sleeping. Additionally, downloadable tools are available to assist with testing software’s resiliency to sleep transitions.

# Availability Requests

To achieve Energy Star rating, today’s PCs ship with automatic sleep and display power management enabled. These features were originally used primarily to increase the battery runtime for mobile PCs. Now a growing number of organizations and individuals utilize these functions to save power on all types of PCs and to increase the lifespan of LCD monitors.

To ensure that these functions work predictably, applications, services, and drivers must correctly use availability requests and ensure that indirect uses of availability requests do not inadvertently cause the display to remain awake or prevent the system from automatically entering sleep.

Applications should use availability requests to:

* Temporarily disable display power management.
* Temporarily disable automatic sleep.
* Enable away mode.

Developers should use availability requests only when they are essential to complete a user scenario. Users expect automatic sleep or display power management to function as they enabled it in Power Options. If an availability request disables one of these power management features, users are unlikely to know which application or driver caused this condition.

Developers should create availability requests when the user scenario begins and release it promptly when the user scenario is completed. Testers must validate the use of availability requests to ensure that power management features are disabled only when necessary. Validation should ensure that the availability request is created when the user scenario begins and is promptly released when the user scenario is completed. In particular, it is extremely important to validate that the use of an external component does not create a long-lived availability request when one is not needed.

A common example of an indirect availability request keeping the PC from automatically sleeping is an application that renders audio for long periods of time. The Windows audio stack raises an availability request whenever an audio channel is open. Therefore, applications that do not close the audio stack when they are not actively playing audio (such as when audio is paused) prevent the PC from automatically sleeping. This is particularly true of systems that have USB sound cards. Windows attempts to automatically close the audio stream for PCI-based and motherboard-integrated sound cards, but Windows 7 does not do this for USB-based sound cards. Therefore, it is advisable to manually close the sound channel when audio is paused or finishes playing.

This paper covers high-level guidelines for the use of availability requests. For full documentation on how to implement availability requests, see “Power Availability Requests “ on the WHDC website.

## Best Practices for Availability Requests

Developers of applications, services, and drivers can use availability requests to temporarily disable display power management, to temporarily disable automatic sleep, and to enable away mode for entertainment and media PC scenarios. The following best practices apply when developers implement availability requests:

* Use **PowerCreateRequest** and related functions in user-mode code if it is possible. Use **GetProcAddress** to determine whether **PowerCreateRequest** is available on the running operating system. If the system does not support this function, use **SetThreadExecutionState** if it is required.
* Set the availability request by using **PowerSetRequest** only when the availability request is required. Clear the request by calling **PowerClearRequest** as soon as the scenario is completed.
* Provide a localized, textual reason for the availability request when you call **PowerCreateRequest** to create the request context.
* Clean up all request objects and associated handles before process exit or service stop.
* Enable away mode only for entertainment and media PC scenarios.

## Best Practices for Scenarios that Involve Passive Viewing of Content

Previously, developers of video applications such as DVD playback software contended with display blanking time-outs. Windows 7 introduced automatic screen dimming functionality. Now, mobile PC users are discovering that their display dims when they watch even short videos or presentations or when they participate in a video chat. This occurs because users are not actively interacting with the PC while they watch the video and because display dimming time-outs are much shorter than typical display blanking or system sleep time-outs.

To avoid requiring users to increase the display dimming time-out or disable it altogether, all applications that render content that is intended for passive viewing must make explicit availability requests for the display to stay awake and not dim while the content renders. Additionally, an application should drop the availability request when it is minimized or otherwise not visible (such as in an inactive Microsoft® Internet Explorer tab).

Generally, applications that display animations or similar looping visual experiences should not create availability requests on the display unless users explicitly put the application into full-screen mode. Automatic screensavers should never raise an availability request unless users launch the screensaver explicitly or directly (such as for use in a photo display mode).

The prevalence of LCD monitors means that screensavers are no longer required to prevent burn-in and actually use more energy than simply allowing the screen to blank.

Also, to conserve system power, applications that have computationally intensive display rendering should register for power events to determine whether the display is turned off. Developers can consider slowing down the frame rate at which applications render or totally disable rendering video until displays turn on again. This technique can also be used if the application is minimized or otherwise not visible. For further information, see “Registering for Power Events” on the MSDN® website.

## Best Practices for Long-Running Processes

Developers who create processes that must run uninterrupted for long periods of time (such as greater than a few minutes) must carefully consider whether they should employ system availability requests.

In particular, if data loss could occur if the system enters automatic sleep (such as recording TV or downloading or copying data) or if the user experience is seriously undermined by a data set not being available (such as an initial index for an application), applications should use an availability request to postpone automatic sleep until the job is completed. Developers should also be careful to clear the availability request as soon as the process is completed and not wait for user confirmation before doing so because it may be hours or even days before the user returns to the PC.

## Using Availability Requests When on Battery Power

Applications that perform long-running tasks and raise a corresponding availability request should register for power events and react accordingly. For details on how to do this, see “Registering for Power Events” on the MSDN website.

Developers should design applications to postpone tasks or warn users not to start a task if it is likely to take longer to complete the task than the remaining charge of the battery. In particular, for lengthy tasks that cannot be interrupted (such as burning a DVD), the application should advise users to provide a constant power source. If an application starts while the system is running on AC power and is then switched to battery, users should be warned that the application could require more energy (watt-hours) than is available in the battery.

## “Energy Smart” Application Summary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Activity | System availability request | Display availability request | Power source aware | Calculate battery charge remaining | Disable/ slow down rendering if monitor off or window hidden |
| Render video content | No | Create | Not applicable | Not applicable | Yes |
| Pause-stop video content | No | Clear | Not applicable | Not applicable | Not applicable |
| Long-running batch process | Create | No | Yes | Yes | Not applicable |

## Availability Requests Diagnostic Tools

PC administrators, developers, and testers can use the new options in the PowerCfg utility in Windows 7 and Windows Server® 2008 R2 to identify problems with power management, including display power management and automatic sleep. The following best practices apply for administrative management of availability requests:

* Use PowerCfg with the **/requests** option to identify outstanding availability requests that prevent display power management or automatic sleep. Run PowerCfg from an elevated command prompt.
* Use PowerCfg with the **/requestsoverride** option to override availability requests. Override only the minimally required set of requests to achieve the desired system behavior. Run PowerCfg from an elevated command prompt.
* Distribute PowerCfg commands to each system in the enterprise by using a Group Policy script.

# Supporting Sleep Transitions

If a sleep or hibernate transition takes too long or is unreliable, users and organizations might be required to entirely disable sleep (automatic and manual) so that they do not lose productivity. This means they will not benefit from the potential power and economic savings that sleep mode provides. As a result, customers may consider not using applications that interfere with sleep mode.

The sleep transition involves two phases: suspend and resume. The following sections provide more detail about these two phases.

### Suspend Phase

During the suspend phase, Windows saves system context and turns off the devices that are required to enter the target state—either sleep or hibernate. Figure 1 shows the subphases of the suspend phase during sleep and hibernate transitions.

Figure 1. Subphases of the suspend phase for sleep and hibernate transitions

During a sleep transition, suspend involves the following subphases:

* **SuspendApps.** Suspend notifications (WM\_POWERBROADCAST messages that have an event type of PBT\_APMSUSPEND) are serially sent to all applications. For each application, the system sends the WM\_POWERBROADCAST message, waits up to 2 seconds for the application to return from processing the message, and then sends the message to the next application. After Windows notifies all applications, it continues to the SuspendServices subphase. Any application that fails to complete its work before the 2-second time-out can continue any remaining processing after the system resumes from sleep.
* **SuspendServices.** All services that registered to receive power management events receive suspend notifications.
* **QueryDevices.** All device drivers receive a power I/O request packet (IRP) that has the IRP\_MN\_QUERY\_POWER minor IRP code and requests a system power state. This request notifies the driver that the system power state is about to change and gives the driver the opportunity to prepare for the change. Most drivers do very minimal work to process this IRP because the typical duration of this subphase is less than 20 milliseconds.
* **SuspendDevices.** All device drivers receive a system power IRP that has the IRP\_MN\_SET\_POWER minor IRP code and request a lower power state. Device drivers must then save appropriate device context and put the device in the appropriate state for sleep or hibernate.

A hibernate transition involves the following additional subphase:

* **HiberfileWrite.** This subphase applies only to hibernate and hybrid sleep transitions. Hybrid sleep, which was introduced in Windows Vista®, combines sleep and hibernate. The system enters sleep after it saves a minimal system context to disk. This allows the system to be restored even if an unexpected power failure occurs. The Hiberfile is written to disk before the system enters sleep.

### Resume Phase

During the resume phase, the system returns to power state S0, the working state. System context is restored to memory, and devices are restarted. The subphases of the resume phase differ, depending on whether the system is resuming from sleep or from hibernate. Figure 2 shows the subphases of the resume phase for the sleep transition.



Figure 2. Subphases of the resume phase for sleep transition

Resume from sleep involves the following subphases:

* **BIOSInitialization.** When a system resumes from sleep (S3 state), the first subphase is BIOS initialization. In this subphase, the platform firmware restores the processor and chipset state.
* **ResumeDevices.** The Windows power manager notifies drivers that the system is resuming by sending system set-power IRPs to all device stacks. This subphase is considered complete when all drivers complete the system set-power IRPs.
* **ResumeApps.** The system sends to each application a WM\_POWERBROADCAST message with the *wParam* event value of PBT\_APMRESUMEAUTOMATIC. User-mode applications might also receive this message with the PBT\_APMRESUMESUSPEND value if a user-initiated action, such as pressing a power button, triggered wake.
* **ResumeServices.** The power managercalls an interface in the user-mode PnP manager, which then calls each service’s power management event handler.
* **PostResume.** The PostResumesubphase is a conceptual subphase that encompasses the time that follows the resume transition until the system returns to a fairly idle state.

As Figure 3 shows, resume from the hibernate transition includes the POST subphase instead of BIOSInitialization and adds the HiberfileRead subphase. All other phases are the same as in resume from sleep.

Figure 3. Subphases of the resume phase for hibernate transition

The POST and HiberfileRead subphases are as follows:

* **POST.** When a system resumes from hibernate (ACPI S4 state), the first subphase is POST (power-on self test).
* **HiberfileRead.** During HiberfileRead, drivers in a special driver stack read the Hiberfile into memory because the normal disk device stack has not yet been loaded. This subphase applies only to hibernate and hybrid sleep transitions.

## Sleep/Hibernate Transition Recommendations for Application Developers

* Avoid WM\_POWERBROADCAST processing delays.

During the SuspendApps subphase, Windows serially sends to all applications a WM\_POWERBROADCAST message with an event type of PBT\_APMSUSPEND. After the system sends the WM\_POWERBROADCAST message, it waits up to 2 seconds for the application to return from processing the message before it sends the message to the next application. The system then continues the suspend process. Any applications that do not complete all their work before the time-out must continue any remaining processing after the system resumes from sleep.

Because of the serial notification, every application that does not respond quickly to the WM\_POWERBROADCAST message blocks the suspend path and directly adds to the overall suspend time on the system. Applications do not always have sufficient CPU time to complete their work before the 2-second time-out elapses. Because of this, several other processes might run concurrently on the system and consume CPU resources.

* Minimize post-resume activity.

During the ResumeApps subphase, Windows sends to all applications a WM\_POWERBROADCAST message with the *wParam* event value of PBT\_APMRESUMEAUTOMATIC. User-mode applications might receive this message with the PBT\_APMRESUMESUSPEND value if a user-initiated action, such as pressing a power button, triggered wake. These notifications are asynchronous, and the kernel power manager considers this subphase complete after it notifies the USER subsystem.

Follow these general guidelines when handling these events in an application:

* Maintain awareness of connectivity status and respond reasonably and transparently to transitions in network connectivity. Automatically reconnect to network resources in the background without interrupting users.
* Manage resource usage immediately after a resume-from-sleep transition. Several applications can respond to resume events at the same time, and users might attempt to quickly access the calendar or other data. Applications that perform resource-intensive system scans should wait until the system is idle before they start such scans.
* If tracking user presence is critical to the application, update this user information immediately upon resume. Use the PBT\_APMRESUMESUSPEND event to determine whether the user is present at the system.

To improve system responsiveness, applications should minimize CPU and disk utilization during the post-resume subphase. By analyzing the CPU and Disk Utilization graphs and comparing them against valid baseline measurements, developers can identify anomalous CPU and disk activity during this subphase.

For more details on power management events and measuring system resume performance, see “Application Power Management Best Practices for Windows Vista” on the WHDC website.

## Sleep/Hibernate Transition Recommendations for Service Developers

* Avoid registering to receive suspend notifications.

If a service is not required to complete any significant work during the suspend phase, it should not opt in to receive suspend notifications. Windows serially sends these notifications to services, and each such notification adds to the overall suspend time of the system.

A driver that defers any required work until the resume notification and then performs it in the background without interrupting users helps to minimize suspend times.

* Minimize post-resume activity.

During the ResumeServices subphase, the kernel power manager calls an interface in the user-mode PnP manager, which then calls each service’s power management event handler to notify the service that the system has resumed to the working state. These notifications are asynchronous, and the kernel power manager considers this subphase complete as soon as it finishes its call to the user-mode PnP manager. As a result, services should follow these guidelines to ensure a satisfying post-resume user experience:

* Optimize CPU and disk usage immediately after the system resumes. During the post-resume phase, users can interact with the system and might try to quickly access the calendar or other data at the same time that several services and applications respond to resume events. By analyzing the CPU and Disk Utilization graphs, as well as comparing the graphs against valid baseline measurements, developers can identify anomalous CPU and disk activity during the ResumeServices subphase.
* Consider using the task scheduler or trigger start mechanism for periodic service work such as software updates so that these tasks do not start immediately after resume.
* If the service depends on a network, maintain awareness of connectivity status and respond reasonably and transparently to transitions in network connectivity.

## Sleep/Hibernate Transition Recommendations for Driver Developers

* Implement fast resume.

Windows Development Foundation (WDF) automatically handles fast resume for kernel-model driver framework (KMDF) and user-mode driver framework (UMDF) drivers.

To achieve fast resume performance in a Windows Driver Model (WDM) driver, follow these guidelines. The guidelines apply only to drivers for leaf node devices—that is, devices that have no child devices in the device hierarchy:

1. Set an I/O completion routine for the IRP\_MN\_SET\_POWER request for the S0 system power state.

2. Send the IRP\_MN\_SET\_POWER IRP for the S0 system power state down the stack.

3. In the I/O completion routine for the S0 system power state IRP\_MN\_SET\_POWER IRP:

* Send an IRP\_MN\_SET\_POWER request for the device D0 power state.
* Return STATUS\_SUCCESS from the I/O completion routine for the S0 system power state IRP\_MN\_SET\_POWER request.
* Complete any device initialization that is required to bring the device online in the I/O completion routine for the D0 set-power IRP.

4. Begin handling any queued I/O requests after the device has returned to the D0 state.

In addition, remember the following guidelines when you optimize a driver for fast resume:

Windows records “resume complete” after all devices have completed the S0 set-power IRPs. Devices are not required to complete their D0 set-power IRPs or to be fully functioning at this point.

For better resume performance, drivers can complete the S0 set-power IRP before the D0 set-power IRP is completed. The power manager can then send the S0 set-power IRP to other devices sooner, which reduces serialization delays and enhances overall system resume performance.

To ensure the best user experience, drivers should queue any I/O requests that they receive while they process the D0 set-power IRP. Failing I/O requests during this time might cause applications to hang or produce time-out error messages.

* Optimize CPU resource usage on resume.

To enhance the post-resume user experience, drivers can optimize use of CPU resources when the system resumes from sleep or hibernate. The performance of deferred procedure call (DPC) routines and other code that runs at IRQL >= DISPATCH\_LEVEL is an important consideration.

Excessively long DPC routines can block other threads from executing and can block other DPC routines that are queued and ready to execute. When threads or DPCs are blocked, the duration of the device initialization phase increases and extends the overall system resume time. The best resume times are achieved when individual DPC routines do not execute for more than 100 µs. If a task must run at DISPATCH\_LEVEL for longer than 100 µs, the DPC routine should terminate after 100 µs and schedule one or more DPC timer routines to complete the task later.

## Sleep/Hibernate Transition Recommendation for PC System Manufacturers

* Optimize S4 resume by enhancing BIOS INT13 performance.

Manufacturers can optimize performance in the HiberfileRead subphase by carefully analyzing and enhancing BIOS INT13 performance. Windows uses these routines for disk I/O to read the hibernate image file from the disk during S4 resume. Consider the following optimizations:

Tune INT13 performance for maximum read performance.

Enable the disk controller for the highest available direct memory access (DMA) mode in the INT13 calls.

Optimize the INT13 calls for large read requests (64 KB).

Optimize for sequential sector reads, that is, multiple 64‑KB reads that are issued in a row, if requests are physically adjacent.

# Summary

With increasing numbers of individuals and organizations enabling automatic sleep and display power management, it is important for application, services, and driver developers to ensure that their products work well and correctly in an environment where systems and displays are entering sleep automatically.

Developers must ensure that their software is resilient to sleep/resume transitions and must not unnecessarily prevent automatic sleep and display power management from working correctly. Additionally, products that consider the energy source or limitations that the system must contend with are valuable to both end users and IT administrator. Over the longer term, software products that are not “energy smart” are at risk of being excluded from installation on corporate PCs and even PCs that individuals own.

# Resources

WHDC website:

Application Power Management Best Practices for Windows Vista

<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PM_apps.mspx>

Energy Smart Software

<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/Energy-Smart_SW.mspx>

Power Availability Requests

<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/AvailabilityRequests.mspx>

Timers, Timer Resolution, and Development of Efficient Code

<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/Timer-Resolution.mspx>

MSDN website:

Registering for Power Events

<http://msdn.microsoft.com/en-us/library/aa373195(VS.85).aspx>