Mobile Battery Solutions Guide for Windows Vista

Version 1.1 — February 21, 2008

Abstract

This guide outlines issues and solutions for advancing battery life for mobile PCs that run the Windows Vista® operating system. It provides system design, power policy guidelines, and preinstallation configuration recommendations as well as test configuration recommendations.

This guide is intended for system and component designers, driver developers, test managers, and preinstallation image engineers who design, deliver, and maintain mobile PC products that deliver a great customer experience with Windows®.

This guide also provides useful information for enterprise information technology (IT) professionals who need technical information to support hardware investment and management decisions for their mobile PC users.

For the latest information, see:
 <http://www.microsoft.com/whdc/system/pnppwr/mobilepwr.mspx>

Feedback

* Please provide feedback on the technical rationale, implementation details, case studies, additional tools, and other information that you’d like to see from Microsoft to help appropriately address issues with mobile battery life.
* To provide feedback, enter your comments in the following form on Microsoft® Connect. This form is anonymous, but you must log on to Microsoft Connect:
<http://connect.microsoft.com/Survey/Survey.aspx?SurveyID=5298&SiteID=221>

Disclaimer: This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

© 2008 Microsoft Corporation. All rights reserved.

Microsoft, MSDN, Excel, Outlook, SuperFetch, Visual Studio, Win32, Windows, Windows Media, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

| Date | Change |
| --- | --- |
| February 21, 2008 | Deleted text in Part 2 that indicates wake support is not available for UMDF USB drivers. WInUSB supports this feature for such drivers.  |
| February 6, 2008 | Version 1 publication  |

Contents

[INTRODUCTION: Why We Need Battery Life Solutions 6](#_Toc191352182)

[Platform Elements Affecting Battery Life 6](#_Toc191352183)

[About This Guide 7](#_Toc191352184)

[Recommended Solutions Checklist 8](#_Toc191352185)

[PART 1: Selecting the Hardware Platform for Power Efficiency 12](#_Toc191352186)

[Hardware Platform Selection 12](#_Toc191352187)

[Battery Device 12](#_Toc191352188)

[Processing Platform 13](#_Toc191352189)

[Low-Power Mode Support 14](#_Toc191352190)

[Processor Performance 14](#_Toc191352191)

[Graphics Device and Display 14](#_Toc191352192)

[Firmware 15](#_Toc191352193)

[Firmware Support for Low-Power Sleep Modes 15](#_Toc191352194)

[Firmware Support for Device Power-Management Technologies 16](#_Toc191352195)

[Hardware Platform Recommendations for Extended Battery Life 16](#_Toc191352196)

[Hardware Platform Selection References 17](#_Toc191352197)

[PART 2: Device and Driver Performance and Power 18](#_Toc191352198)

[Introduction: Device and Driver Selection 18](#_Toc191352199)

[Driver Implementation Guidelines 19](#_Toc191352200)

[Driver Activity 19](#_Toc191352201)

[Runtime Idle Detection 20](#_Toc191352202)

[Integration with Windows Power Policy 20](#_Toc191352203)

[Power Setting Notification 20](#_Toc191352204)

[Custom Power Policy Settings 21](#_Toc191352205)

[User-Mode Services and Device-Specific Applications 21](#_Toc191352206)

[Device Class–Specific Information 22](#_Toc191352207)

[PCI Express 22](#_Toc191352208)

[Graphics 23](#_Toc191352209)

[VSync Interrupt 24](#_Toc191352210)

[Graphics and Display Driver Considerations 24](#_Toc191352211)

[USB Hubs and Controllers 24](#_Toc191352212)

[Integrated USB Hubs 25](#_Toc191352213)

[USB Host Controllers 25](#_Toc191352214)

[USB Selective Suspend 25](#_Toc191352215)

[Wireless Devices 28](#_Toc191352216)

[Wireless Device Selection 28](#_Toc191352217)

[Wireless Driver Considerations 29](#_Toc191352218)

[Storage Devices 29](#_Toc191352219)

[Serial ATA Devices with AHCI Chipsets 29](#_Toc191352220)

[USB-Connected Storage Devices 30](#_Toc191352221)

[Disk Idle Detection 30](#_Toc191352222)

[Audio Devices 31](#_Toc191352223)

[Optimization of Audio Hardware Power Consumption 31](#_Toc191352224)

[Optimization of Audio Software and Driver Power Consumption 32](#_Toc191352225)

[Devices and Drivers Recommendations 32](#_Toc191352226)

[Device and Driver Selection References 33](#_Toc191352227)

[PART 3: Operating System Configuration of Power Policy 35](#_Toc191352228)

[Introduction to Power Policy Configuration 35](#_Toc191352229)

[Power Policy Infrastructure 36](#_Toc191352230)

[Power Plans 36](#_Toc191352231)

[Power Policy User Experience Elements 37](#_Toc191352232)

[Optimizing Power Policy for Extended Battery Life 39](#_Toc191352233)

[Display Idle Timeout 39](#_Toc191352234)

[Display Brightness 39](#_Toc191352235)

[Hard Disk Drive Idle Timeout 40](#_Toc191352236)

[802.11 Wireless Power-Save Mode 40](#_Toc191352237)

[Idle Detection Threshold 41](#_Toc191352238)

[SATA AHCI Link Power Management 42](#_Toc191352239)

[Interlaced DVD Playback 42](#_Toc191352240)

[Optimizing Processor Power-Management Policy 43](#_Toc191352241)

[Favoring Power over Performance 43](#_Toc191352242)

[Reducing Maximum Performance State 44](#_Toc191352243)

[Power Policy Recommendations for Extended Battery Life 45](#_Toc191352244)

[Power Policy Optimization References 46](#_Toc191352245)

[PART 4: Preinstalled Software and Energy Efficiency 48](#_Toc191352246)

[Battery Life Impact of Preinstalled Software 48](#_Toc191352247)

[Why Preinstalled Software Matters for Battery Life 48](#_Toc191352248)

[Understanding Resource Usage 50](#_Toc191352249)

[CPU and Power Efficiency 50](#_Toc191352250)

[GPU and Power Efficiency 51](#_Toc191352251)

[Disk I/O Usage and Power Efficiency 51](#_Toc191352252)

[Best Practices for Software Applications 52](#_Toc191352253)

[Guidelines for Power-Efficient Applications 52](#_Toc191352254)

[High-Resolution Periodic Timers and Power Efficiency 53](#_Toc191352255)

[Issues for timeBeginPeriod in Multimedia Applications 53](#_Toc191352256)

[Best Practices for High-Resolution Periodic Timers 54](#_Toc191352257)

[Power Event Monitoring Tool: Displaying Power Event Notices 54](#_Toc191352258)

[Recommended Testing for Clean vs. Preinstallation Images 56](#_Toc191352259)

[Application Power-Management References 57](#_Toc191352260)

[PART 5: Evaluating and Measuring Battery Life 58](#_Toc191352261)

[Getting Started with Measuring Energy Efficiency 58](#_Toc191352262)

[Choosing Scenarios to Measure 60](#_Toc191352263)

[Idle Test Scenario 62](#_Toc191352264)

[DVD Playback Test Scenario 62](#_Toc191352265)

[Applications in Test Scenarios 62](#_Toc191352266)

[Benchmarks and Battery Life 63](#_Toc191352267)

[General Performance and Analysis Tools from Microsoft 64](#_Toc191352268)

[Event Tracing for Windows 65](#_Toc191352269)

[Windows Reliability and Performance Monitor: Viewing Processor Usage 66](#_Toc191352270)

[Xperf: Measuring and Analyzing System Performance 69](#_Toc191352271)

[PwrTest: Viewing Supported PPM States and Transitions 70](#_Toc191352272)

[Testing Tools Reference 70](#_Toc191352273)

[APPENDIX A: Test Methodology for Measuring Battery Life 72](#_Toc191352274)

[Principles for Testing Battery Life 72](#_Toc191352275)

[Overview of Testing Methodology 73](#_Toc191352276)

[Creating a Baseline Configuration for Testing 73](#_Toc191352277)

[Installing a Clean Version of Windows 73](#_Toc191352278)

[Installing Current Windows Updates 74](#_Toc191352279)

[Installing Updated Drivers 74](#_Toc191352280)

[Configuring Hardware and Operating System for Testing 75](#_Toc191352281)

[Disabling Network Connections and External Devices 76](#_Toc191352282)

[Disabling the Screensaver 76](#_Toc191352283)

[Disabling the Display Idle Timeout 76](#_Toc191352284)

[Disabling the Sleep Idle Timeout 77](#_Toc191352285)

[Disabling the Hibernate Idle Timeout 77](#_Toc191352286)

[Disabling Automatic Windows Updates 77](#_Toc191352287)

[User Account Control Elevation Prompts 77](#_Toc191352288)

[Disabling System Restore (Optional) 78](#_Toc191352289)

[Disabling Volume Shadow Copy Service (Optional) 78](#_Toc191352290)

[Executing Idle Tasks 79](#_Toc191352291)

[Scheduled Tasks 79](#_Toc191352292)

[Conditioning and Training the System before Testing 80](#_Toc191352293)

[Running the Battery Life Tests 81](#_Toc191352294)

[Running the Idle Battery-Life Test 81](#_Toc191352295)

[Running the DVD Playback Test 81](#_Toc191352296)

[Evaluating Changes 82](#_Toc191352297)

INTRODUCTION: Why We Need Battery Life Solutions

Windows®-based laptops must meet energy efficiency regulatory requirements such as the United States Environmental Protection Agency (EPA) Energy Star program. Furthermore, surveys have shown that longer battery life continues to be what consumers most want and need in laptops.

To meet consumer demands, Windows laptops must continually advance in the following areas:

* Be more energy efficient in all usage scenarios, including idle, productivity workloads, DVD and media playback, and industry benchmarks.
* Deliver better mobile PC battery life with each product iteration—for hardware platforms and for Windows.

The Windows Vista® operating system—including the advances with Service Pack 1 (SP1)—is proving to be highly reliable and enables fast on-and-off performance. However, extensions provided with mobile PC systems such as services, system tray applets, drivers, and other software can significantly impact performance, reliability, and the user experience.

# Platform Elements Affecting Battery Life

Battery life is a complex problem, with factors affected by and affecting all elements of the PC ecosystem. Small improvements across multiple scenarios can improve battery life, and yet a single poorly-performing application, device, or system feature can significantly reduce the battery life on a system. Optimizing Windows mobile systems for energy efficiency will require continued investments by the entire industry to ensure that platforms are optimally designed and configured. At Microsoft, we are committed to advancing each generation of products in this particular area.

Here are the principle issues and components that affect battery life in a mobile PC:

Battery Capacity

* Size, type and quality of battery capacity affect battery life.
* The larger the battery, the greater the power supply.
* However, larger batteries are more expensive and heavier, whereas users prefer lighter systems.

Hardware Components

* How frequently and deeply can the hardware enter into lower power states?
* Do the hardware components support lower power states?
* Are drivers optimized for energy efficiency?

Operating System–Directed Power Management

* How efficient is Windows code while under a load versus idle?
* Do all components cooperate with Windows-directed power management?
* Has the operating system been properly configured to optimize for power management through power policy settings?

Application Software and Services

* How efficient are applications, drivers, and services while under a load versus idle?
* How well does software cooperate with operating system–directed power management?
* Does the software prevent the system or devices from entering into low-power idle states?

# About This Guide

This guide summarizes knowledge that was gained over months of working with PC manufacturers to identify issues and define solutions for measuring and then optimizing battery life and on-and-off performance on Windows laptops.

This guide assumes that you as a PC professional—whether a system or device designer, driver or software developer, or tester—already consider “improve battery life for mobile PCs” as a top-priority issue. Therefore, this guide dives right into the key details. Two sources provide the foundation for these guidelines:

* Deep knowledge of the Windows and PC architecture issues, directly from the Windows Base and Performance teams at Microsoft.
* Extensive testing and forensics on battery performance and energy efficiency over hundreds of mobile PCs running Windows Vista and Windows XP.

Overall, this guide examines the issues that are related to battery life in the following order:

* Selecting hardware for Windows laptops.
* Optimizing hardware and drivers for Windows laptops.
* Optimizing for operating system–direct power management through power policies.
* Optimizing preinstalled software for energy efficiency.
* Measuring energy efficiency.

We believe that you as a PC professional have specific problems to solve with existing products or want to combine best practices into your new designs. Therefore, this guide presents recommendations for each category in the following format:

* Understanding the issues.
* Examples and details from research.
* A checklist of best practices.

Where significant implementation or conceptual details are required, we will direct your attention to other references to or appendixes in these chapters.

# Recommended Solutions Checklist

Basic Principles. Here are the basic principles for designing components and systems that support maximum battery life:

* Every performance optimization is a battery life optimization.
* Decreased performance is acceptable when the system is on battery or when the user has selected a Power Saver power plan.
* All components in a particular system design must be tested to understand what is affecting power usage in the system—and the systems must be retested after any changes to drivers, codecs, power policy settings, or preinstalled software.

Recommendations. Here’s a summary list of the solutions recommended in this guide:

[**PART 1: Selecting the Hardware Platform for Power Efficiency 13**](#_Toc189996545)

[Battery Device 13](#_Toc189996547)

[Recommendations for System Manufacturers 14](#_Toc189996548)

[ Take advantage of the advice in this guide to optimize the workload power consumption. 14](#_Toc189996549)

[ Provide a suitable battery device to yield the desired battery lifetime. 14](#_Toc189996550)

[Processing Platform 14](#_Toc189996551)

[Recommendation for System Manufacturers 15](#_Toc189996552)

[ Research and understand the power-management capabilities of each specific chipset before implementing the chipset in a mobile PC design. 15](#_Toc189996553)

[Graphics Device and Display 15](#_Toc189996554)

[Recommendations for System Manufacturers 16](#_Toc189996555)

[ Work closely with processor and chipset vendors to select a GPU device with the lowest power consumption possible, whether the GPU is discrete or integrated. 16](#_Toc189996556)

[ Validate that the selected GPU device’s power-management technologies
operate correctly. 16](#_Toc189996557)

[ Consider reducing the maximum performance of a GPU when the system is on battery power or in a Power Saver power plan. 16](#_Toc189996558)

[ Validate the efficiency of graphics device drivers after platform integration. 16](#_Toc189996559)

[Recommendations for System Manufacturers—Mobile PC Displays 16](#_Toc189996560)

[ Consider the use of light emitting diode (LED)-backlit LCD displays for systems where maximum battery life is required. 16](#_Toc189996561)

[ Consider integrating an ambient light sensor (ALS) device to ensure that the
display has minimal impact on overall system power consumption. 16](#_Toc189996562)

[Firmware 17](#_Toc189996563)

[Recommendations for System Manufacturers 17](#_Toc189996564)

[ Enable very fast resume from sleep (ACPI S3 state). 17](#_Toc189996565)

[ Wake the system automatically when the battery capacity is low. 17](#_Toc189996566)

[ Expose a brightness control for the internal display. 17](#_Toc189996568)

[ Support processor performance and idle states. 17](#_Toc189996569)

[Hardware Platform Recommendations for Extended Battery Life 17](#_Toc189996570)

[Recommendations for System Manufacturers 17](#_Toc189996571)

[ Take advantage of chipset and processor components that can dynamically scale their power consumption with performance. 17](#_Toc189996572)

[ Validate that other core chipset power-management technologies are enabled in the mobile PC platform. 18](#_Toc189996573)

[ Implement firmware support for platform power-management technologies. 18](#_Toc189996574)

[**PART 2: Device and Driver Performance and Power 20**](#_Toc189996576)

[Driver Implementation Guidelines 21](#_Toc189996578)

[Recommendations for Device Manufacturers and Driver Developers 21](#_Toc189996579)

[ Eliminate unnecessary driver activity and optimize for performance. 21](#_Toc189996580)

[ Support runtime idle detection for each device. 21](#_Toc189996581)

[ Support device-specific power features, such as device-performance states, when feasible to do so. 21](#_Toc189996582)

[ Integrate device power policy with Windows power policy. 21](#_Toc189996583)

[User-Mode Services and Device-Specific Applications 23](#_Toc189996587)

[Recommendations for Services and Applications Developers 24](#_Toc189996588)

[ Invest in performance optimizations. 24](#_Toc189996589)

[ Adjust to user power policy by reducing resource usage or increasing performance accordingly. 24](#_Toc189996590)

[ Reduce the frequency of periodic background tasks such as updates when the system is on battery power. 24](#_Toc189996591)

[ Avoid polling and spinning in tight loops, which use the processor heavily and reduce the effectiveness of processor idle states and processor
performance states. 24](#_Toc189996592)

[ Do not use high-resolution (<15‑ms) periodic timers, which similarly reduce the efficiency of processor power-management technologies. 24](#_Toc189996593)

[ Respond to common power events, such as power plan personality and system power source (AC/DC) changes, to help extend system battery life. 24](#_Toc189996594)

[PCI Express 24](#_Toc189996596)

[Recommendation for System Manufacturers 25](#_Toc189996597)

[ Validate that each PCIe device supports and uses ASPM. 25](#_Toc189996598)

[Graphics 25](#_Toc189996599)

[Recommendations for Graphics and Display Drivers 26](#_Toc189996600)

[ Support brightness control through the ACPI brightness object. 26](#_Toc189996601)

[ Support disabling VSync for Windows Vista SP1 and later releases. 26](#_Toc189996602)

[ Do not wait in a busy loop in the display driver, which maintains high CPU usage. 26](#_Toc189996603)

[USB Hubs and Controllers 26](#_Toc189996604)

[Recommendations for System Manufacturers 26](#_Toc189996605)

[ Support selective suspend in all USB devices. 26](#_Toc189996606)

[ Avoid integrating USB hubs into mobile platforms. 26](#_Toc189996607)

[Wireless Devices 30](#_Toc189996608)

[Recommendations for System Manufacturers 31](#_Toc189996609)

[ Wireless radios should automatically power off when they are not in use. 31](#_Toc189996610)

[ Wireless devices should support appropriate power settings for each of the Windows power profiles. 31](#_Toc189996611)

[Storage Devices 31](#_Toc189996612)

[Recommendation for System Manufacturers 31](#_Toc189996613)

[ AHCI optical drives should support asynchronous notification (AN), which helps reduce the need for the operating system to poll optical drives for disc insert and ejection status. 31](#_Toc189996614)

[ If a USB-connected storage device is integrated into the system, provide
 additional software and hardware to automatically disconnect and power off
the device when it is not in use. 32](#_Toc189996616)

[Recommendations for Device and System Manufacturers 32](#_Toc189996617)

[ Customize disk idle and burst ignore timeouts for hardware. 32](#_Toc189996618)

[ Set appropriate disk idle detection timeout values and disk burst ignore threshold values in system policies. 32](#_Toc189996619)

[Audio Devices 33](#_Toc189996620)

[Recommendations for System Manufacturers 33](#_Toc189996621)

[ Integrate your audio adapter and miniport drivers with the Microsoft audio port class driver (PortCls.sys). 33](#_Toc189996622)

[ Select and integrate audio devices (codecs) that can enter D3 at runtime without audible pops or clicks on entering D3 or returning to D0. 33](#_Toc189996624)

[ Remove the 8254 timer chip so that the system can play accessibility beeps when the audio device is powered off. 33](#_Toc189996625)

[ Remove the analog CD connection from the CD-ROM/DVD-ROM to the audio
device to avoid requiring the audio device to be powered on indefinitely so that
the user can listen to audio CDs. 33](#_Toc189996626)

[ If possible, avoid polling of S/PDIF input. 34](#_Toc189996627)

[ Use Class D amplifiers for speakers and power them down when speakers are not being used (device is idle). 34](#_Toc189996628)

[ Use High-Definition Multimedia Interface (HDMI) audio hardware that can put
the display into a low-power mode without stopping audio transmission when
the video is not being used. 34](#_Toc189996629)

[ Enable proprietary hardware encryption only when absolutely necessary. 34](#_Toc189996630)

[ Use Windows Vista SP1 to enable key power-management enhancements
for audio devices. 34](#_Toc189996632)

[ Use audio drivers that take advantage of the Windows audio system’s PortCls idle timer functionality (SP1 and later only). 34](#_Toc189996633)

[ Use drivers that monitor the power plan that the user selects. 34](#_Toc189996634)

[Devices and Drivers Recommendations 34](#_Toc189996635)

[Recommendations for System Manufacturers 34](#_Toc189996636)

[ Ensure that all devices implement runtime idle power management. 34](#_Toc189996637)

[ Ensure that the user can disable wireless devices to save power. 34](#_Toc189996638)

[ Incorporate graphics device hardware that can automatically reduce power consumption when the system is on battery power. 35](#_Toc189996639)

[ Validate that device power controls and power states are integrated with Windows power policy. 35](#_Toc189996640)

[ Validate that all integrated USB devices support selective suspend and automatically enter the suspend state when idle. 35](#_Toc189996641)

[ Avoid integrating USB hub devices. USB hub devices draw extra battery power and extend resume performance. 35](#_Toc189996642)

[ Validate selective suspend support for USB-connected Bluetooth radios. 35](#_Toc189996643)

[ Avoid integrating USB devices that do not support selective suspend, including mass-storage devices such as optical drives and USB flash drives. 35](#_Toc189996644)

[ Validate that SATA storage devices support both HIPM and DIPM. 35](#_Toc189996645)

[ Enable DIPM in the software preinstallation when using the Microsoft SATA
storage driver (Msahci.sys). 35](#_Toc189996646)

[ Incorporate audio codec hardware that supports runtime idle power management by supporting the D3 power state. 35](#_Toc189996647)

[ Implement sensor devices such as accelerometers, sensors, and buttons by using
an interrupt model or ACPI notifications to avoid software polling to detect
trip and failure conditions. 35](#_Toc189996648)

[**PART 3: Operating System Configuration of Power Policy 37**](#_Toc189996650)

[Power Policy Recommendations for Extended Battery Life 47](#_Toc189996666)

[Recommendation for System Manufacturers and IT Pros 47](#_Toc189996667)

[ Define appropriate default power policies for both AC and DC power to maximize battery life. 47](#_Toc189996668)

[**PART 4: Preinstalled Software and Energy Efficiency 50**](#_Toc189996670)

[Understanding Resource Usage 52](#_Toc189996673)

[Recommendation for System Manufacturers—CPU and Power Efficiency 53](#_Toc189996674)

[ Target 98-percent average processor idle time over a 10‑minute period of idle. 53](#_Toc189996675)

[Recommendation for System Manufacturers—GPU Resource Usage 53](#_Toc189996676)

[ Follow the same guidance as for CPUs. 53](#_Toc189996677)

[Recommendation for System Manufacturers—Disk I/O 53](#_Toc189996678)

[ Ensure that the software in the preload image allows the disk to spin down for at least 75 percent of an idle period one hour or greater. 53](#_Toc189996679)

[Guidelines for Power-Efficient Applications 54](#_Toc189996681)

[Recommendations for Application Developers 54](#_Toc189996682)

[ Invest in performance optimizations. 54](#_Toc189996683)

[ Adjust to user power policy. 54](#_Toc189996684)

[ Reduce resource usage when the system is on battery power. 54](#_Toc189996685)

[ Do not render to the display when it is off. 54](#_Toc189996686)

[ Avoid polling and spinning in tight loops. 54](#_Toc189996687)

[ Do not prevent the system from turning off the display or idling to sleep. 54](#_Toc189996688)

[ Respond to common power-management events. 54](#_Toc189996689)

[ Avoid use of high-resolution periodic timers. 54](#_Toc189996690)

[High-Resolution Periodic Timers and Power Efficiency 55](#_Toc189996691)

[Recommendations for Application Developers 56](#_Toc189996692)

[ Understand the impact of high-resolution periodic timers on system
power consumption. 56](#_Toc189996693)

[ Validate that your application does not unexpectedly change the system
timer interval. 56](#_Toc189996694)

[ If your application must use a high-resolution periodic timer, enable the periodic timer only while the required functionality is active. 56](#_Toc189996695)

[ If your application must use a high-resolution periodic timer, consider disabling use of the periodic timer and associated functionality when a Power Saver power plan is active or when the system is on battery power. 56](#_Toc189996696)

[Recommended Testing for Clean vs. Preinstallation Images 58](#_Toc189996698)

[Recommendation for System Manufacturers and IT Pros 58](#_Toc189996699)

[ Use the testing and analysis tools to understand which activities in the preinstallation image might cause the CPU, GPU, and hard drive to consume unnecessary power. 58](#_Toc189996700)

PART 1:
Selecting the Hardware Platform for Power Efficiency

In this part

Hardware Platform Selection

Battery Device

Processing Platform

Graphics Device and Display

Firmware

Hardware Platform Recommendations for Extended Battery Life

Hardware Platform Selection References

# Hardware Platform Selection

Hardware platform components form the basis for battery life in a mobile PC. These components include the battery, processor, core chipset, graphics device, display, and firmware. Each of these hardware components has a critical impact on system power consumption and battery life.

We encourage system manufacturers to follow the recommendations in this part of the guide to understand the impact of each hardware component on battery life. We are not recommending specific hardware manufacturers’ products or technologies. When selecting and integrating hardware components for a particular system design, you should consider the impact on battery life in addition to marketing requirements and component costs.

## Battery Device

The battery device is the basic limiting factor for usable battery life time of a mobile PC. The battery device supplies the electrical power for the system to operate when disconnected from utility power. The battery device has a finite capacity, and this capacity is depleted over time when the mobile PC is in use or in a low-power sleep state.

You can determine battery lifetime when you know both the battery device capacity and the power consumption of the user workload. The basic formula is:

 BatteryLifeTime (Hours) = Battery Capacity (Wh) / Workload Power Rate (W)

The formula shows why most of this guide discusses ways to reduce the workload power rate. Although the battery capacity is fixed on any particular mobile PC, you can find many opportunities to reduce the workload power rate. These reductions can come from more power-efficient hardware devices and more efficient use of system resources by preinstalled applications.

The battery device has a finite full capacity, which is closely related to several key properties:

* Battery chemistry: The type of chemicals used in the battery to store charge.
* Battery size and cell count: The number of individual battery cells within the battery pack device. A bigger battery (and hence more capacity) can be obtained by increasing the cell count.
* Battery weight: Advanced battery chemical technologies and increased battery size and cell count. These properties tend to increase the overall weight of the battery.
* Battery device cost: Batteries with larger capacities. Large batteries almost always have a larger manufacturing cost.

Battery technologies are improving incrementally year by year, with limited prospects for a new battery technology to revolutionize the basic properties of battery capacity, size, and weight. System manufacturers are encouraged to select battery devices that meet the user needs of the mobile PC under design. For example:

* Consider smaller capacity and lighter weight batteries for ultramobile and ultraportable PC systems.

Typically, reduced capacity and weight allow these form factors to be smaller, lighter, and thinner. The downside is that it becomes even more critical to reduce the workload power rate to achieve long battery life on a system with a smaller battery capacity.

* Use high-capacity batteries for thin-and-light systems where the user requires or desires long battery life.

To provide for all-day battery life for enterprise and business users, consider supplying a battery with extended capacity. You can achieve 8-hour battery life with a battery capacity of 80 watt-hours (Wh) and an average workload power consumption of 10 watts (W).

Ultimately, battery device selection for a particular mobile PC is governed by marketing requirements for battery runtime, system weight, and cost.

##### Recommendations for System Manufacturers

* Take advantage of the advice in this guide to optimize the workload power consumption.
* Provide a suitable battery device to yield the desired battery lifetime.

## Processing Platform

The processing platform consists of the processor device and the core chipset. These components have critical impact on battery life for two primary reasons:

* The processor and core chipset are powered on during the entire time that the system is running (in the ACPI S0 state).
* The processor typically has a wide dynamic range of power consumption, from very small when the processor is idle to potentially very large when the processor is under heavy usage.

### Low-Power Mode Support

The most important considerations in selecting processing platform components for long battery life are processor and chipset support for low-power modes. Modern mobile PC processors support two industry-standard processor power management (PPM) capabilities:

* Processor performance (P) states

P-states are discrete processor frequency and voltage combinations. Each discrete combination has different power consumption and performance characteristics.

* Processor idle (C) states

C-states are low-power modes, entered by the processor when idle in the very small amounts of time between the system clock interrupt—typically 15.6 milliseconds (ms)—and other platform break events.

Important: Although a given processor’s individual performance and idle state capabilities might differ, it is critical for maximum system battery life that the processor support multiple performance and idle states.

### Processor Performance

In addition to basic PPM capabilities, overall processor performance should be considered with respect to battery lifetime goals for the whole system. For example, many ultraportable systems are shipped with low-voltage or performance-reduced versions of processors. These systems favor reduced power consumption over maximum system performance. We encourage system manufacturers to consider these design trade‑offs when low power consumption (and hence long battery life) is more important than maximum system performance and responsiveness.

Most chipsets support low-power modes and power-management capabilities. However, these capabilities are not standardized and typically vary by manufacturer.

##### Recommendation for System Manufacturers

* Research and understand the power-management capabilities of each specific chipset before implementing the chipset in a mobile PC design.

## Graphics Device and Display

Graphics devices and displays are often selected for a system based on the manufacturer and capabilities of the core chipset and processing platform. The increasing capabilities and power consumption of modern graphics devices are substantial, so graphics and display should be considered as part of the basic hardware platform, rather than as discrete devices.

With respect to graphics, the following hardware components are primarily considered in terms of their impact on battery life:

* Graphics processing unit (GPU)

The GPU can be integrated into the core chipset or can be a separate (discrete) component.

* Display

The most common mobile PC display technologies are liquid crystal displays (LCDs). The LCD display in a mobile PC can consume up to 40 percent of the workload power consumption when the display is powered on.

Unlike processor devices, GPU power-management technologies are not well standardized in the industry. However, each graphics device manufacturer supports GPUs with low-power modes that favor battery life.

##### Recommendations for System Manufacturers

* Work closely with processor and chipset vendors to select a GPU device with the lowest power consumption possible, whether the GPU is discrete or integrated.
* Validate that the selected GPU device’s power-management technologies operate correctly.
* Consider reducing the maximum performance of a GPU when the system is on battery power or in a Power Saver power plan.
* Validate the efficiency of graphics device drivers after platform integration.

Because GPU power-management technologies tend to be proprietary, the graphics driver is responsible for enabling and managing these technologies. We have observed graphics drivers that have a large impact on GPU device power consumption and overall system resource usage. Therefore, careful evaluation is required for maximum battery life, as described in [Part 4](#Part4) of this guide.

##### Recommendations for System Manufacturers—Mobile PC Displays

* Consider the use of light emitting diode (LED)-backlit LCD displays for systems where maximum battery life is required.

LED-backlit LCD displays typically consume less power than equivalent-sized fluorescent-backlit displays.

* Consult with your GPU vendors about incorporating energy-saving technologies in your platform.

Some GPU and display vendors have proprietary technologies that reduce display subsystem power without noticeably impacting display quality.

* Evaluate different display hardware because power consumption can vary significantly by manufacture and product.
* Consider integrating an ambient light sensor (ALS) device to ensure that the display has minimal impact on overall system power consumption.

The ALS device and associated software allow the display backlight to be adjusted optimally to ambient light. In environment that has relatively low ambient light, this helps reduce power consumption and extend battery life by providing the optimal viewing experience with minimal display brightness.

## Firmware

System firmware (BIOS)—although technically not hardware—plays a critical role in reducing power consumption by enabling many core processing platform and device power-management technologies. Firmware is often responsible for managing and communicating battery device status and for sleeping and waking the system. For these reasons, firmware can directly impact battery life.

### Firmware Support for Low-Power Sleep Modes

Firmware is critical in helping transition the mobile PC to and from low-power sleep states. The system can enter sleep when the user has finished working and closes the mobile PC lid. The system might wake from sleep when the lid is opened or if the system is very low on battery power.

##### Recommendations for System Manufacturers

System manufacturers should validate that the firmware on their mobile PC systems has the following abilities:

* Enable very fast resume from sleep (ACPI S3 state).

The BIOS initialization phase of resume from sleep should be less than 500 milliseconds (ms).

* Wake the system automatically when the battery capacity is low.

The platform should automatically wake when the remaining capacity is just enough to put the system in the hibernate (ACPI S4) state. Windows will automatically detect the low battery capacity and hibernate the system.

### Firmware Support for Device Power-Management Technologies

Firmware is also critical for ensuring correct operation of platform and device power-management technologies. For example, ACPI firmware is responsible for reporting processor power-management capabilities to Windows and for exposing a brightness control for the display.

##### Recommendations for System Manufacturers

System manufacturers should validate that the firmware on their mobile PC systems can correctly:

* Expose a brightness control for the internal display.

ACPI firmware should implement the \_BCL and \_BCM methods under the integrated display device object to allow Windows to control display brightness. When the system transitions to battery power, Windows will reduce display brightness.

* Support processor performance and idle states.

ACPI firmware must accurately describe the processor performance and idle state capabilities. Windows performs stringent validation of processor power-management firmware descriptions and will disable processor performance and idle states if errors are found in the firmware data.

# Hardware Platform Recommendations for Extended Battery Life

##### Recommendations for System Manufacturers

* Take advantage of chipset and processor components that can dynamically scale their power consumption with performance.
* Select a processor that supports multiple ACPI P-states.
* Select a processor that supports ACPI C-states.
* Validate that other core chipset power-management technologies are enabled in the mobile PC platform.

For maximum battery life, take advantage of the latest graphics device display power-management technologies:

* Integrate ALS devices and provide firmware and software support for integrating ALS control with Windows.
* Incorporate display power conservation technologies that do not noticeably impact display quality.
* Use LED-backlit displays to reduce overall system power consumption when the display is on.
* Implement firmware support for platform power-management technologies.
* Enable Windows control of processor performance and idle states.

Validate use of processor power-management technologies by using PwrTest, as described in [Part 5](#Part5) of this guide.

* Design firmware for resume from sleep (ACPI S3) performance.

BIOS initialization times should be less than 500 ms as observed by the PwrTest utility.

* Enable Windows to control display brightness through ACPI controls.

This allows the display brightness to automatically decrease when the system goes on battery power.

* Implement firmware to help reduce sleep state (ACPI S3) power consumption.

Target days and weeks of S3 battery life on a full battery charge.

* Implement hardware and firmware support for wake on critical battery capacity.

This allows the system to wake from sleep and Windows will automatically enter the hibernate state to save the user’s open programs and data.

* Design system firmware to prevent the system from waking on timer alarm when the system is on battery power.

This helps prevent the system from waking up accidentally in an enclosed environment such as a briefcase or a laptop bag.

# Hardware Platform Selection References

White papers on the WHDC Web site:

Application Power Management Best Practices for Windows Vista
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PM_apps.mspx>

Brightness Control in WDDM
<http://www.microsoft.com/whdc/device/display/aero/brightness.mspx>

Measuring System Resume Performance on Windows Vista
<http://www.microsoft.com/whdc/system/sysperf/resumeperf.mspx>

Optimizing Windows Vista Platforms for Energy Efficiency
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/optimize_Power.mspx>

Processor Power Management in Windows Vista and Windows Server 2008
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/ProcPowerMgmt.mspx>

PART 2:
Device and Driver Performance and Power

In this part

Introduction: Device and Driver Selection

Driver Implementation Guidelines

Driver Activity

Runtime Idle Detection

Integration with Windows Power Policy

User-Mode Services and Device-Specific Applications

Device Class–Specific Information

PCI Express

Graphics

USB Hubs and Controllers

Wireless Devices

Storage Devices

Audio Devices

Devices and Drivers Recommendations

Device and Driver Selection References

# Introduction: Device and Driver Selection

Part 2 of this guide provides information on how devices and drivers contribute to energy-efficient mobile platforms. It provides guidelines for implementing drivers and device service applications and supplies device-class specific information about device and driver features that are desirable and those to avoid.

Careful selection of devices and drivers is a critical step in creating an energy-efficient laptop with extended battery life. Every device on the system requires power to operate, and some devices additionally affect the power consumption of other devices or components.

For example, when a USB wireless radio is in the working state, the radio device consumes power, as does the USB controller into which it is connected. A USB device also increases the power consumption of the processor when it is active, due to frequent direct memory access (DMA) transactions by the USB host controller. However, if the device supports selective suspend, the USB device can be powered down when the radio is not in use. Powering down the USB device alleviates the increased power consumption, helping to reduce overall system power consumption.

The selection and implementation of device drivers are just as important as the hardware device itself. Microsoft studies have shown that the battery life on a system can change drastically with a single driver update. Drivers that use excessive system resources, such as consuming CPU time in polling loops or servicing unnecessary interrupts, can degrade overall battery life.

All devices and their drivers should support runtime idle detection. This means that the device automatically turns off or enters a low-power mode when it is inactive and the system is operational. For example:

* Disk drives spin down after an idle timeout.
* USB devices enter selective suspend when idle or after a timeout.

The driver for each device must implement the required support for runtime idle detection, and vendors should test each platform configuration to ensure that all configured devices work together properly.

When the system is on battery power or the user has selected a Power Saver power plan, decreased performance is an acceptable trade-off for reduced power consumption.

# Driver Implementation Guidelines

A well-designed driver contributes to an energy-efficient platform.

##### Recommendations for Device Manufacturers and Driver Developers

* Eliminate unnecessary driver activity and optimize for performance.
* Support runtime idle detection for each device.
* Support device-specific power features, such as device-performance states, when feasible to do so.
* Integrate device power policy with Windows power policy.

## Driver Activity

Driver software activity can prevent the processor from entering a low-power idle (C) state. Drivers should avoid unnecessary activity and optimize for performance by using the following guidelines:

* Implement performance optimizations, such as using fewer processor cycles to perform a given task.
* Avoid stalling the processor at high interrupt request level (IRQL).

Use KeDelayExecutionThread wherever required to wait on the hardware to complete an operation.

* Minimize the use of polling. If possible, use an interrupt instead.
* Profile the driver’s processor activity by using the XPerf suite of tools.

For more information, see [Part 5](#Part5) of this guide.

Device manufacturers and system original equipment manufacturers (OEMs) should consider the trade-offs between power consumption and performance. When the system is on battery power or the user has selected a Power Saver power plan, decreased performance is an acceptable trade-off for reduced power consumption. Drivers can use Windows power policy notifications to automatically change their power consumption and performance behaviors, as described later in Part 2.

## Runtime Idle Detection

Drivers should support runtime idle detection (also called S0 idle) so that they can transition their idle devices to a lower power state while the system is in the on (ACPI S0) state.

Techniques for implementing runtime idle detection vary based on the driver model and the type of device:

* Drivers for USB devices can use USB selective suspend regardless of the driver model.

For details, see “[USB Selective Suspend](#_USB_Selective_Suspend)” later in Part 2.

* Kernel-mode driver framework (KMDF) drivers for other device types can use framework mechanisms.

For details, see “Plug and Play and Power Management in WDF Drivers” on the WHDC Web site. [*[source]*](#_Device_and_Driver)

Note: Sources for all references are listed at the end of each part in this guide.

* User-mode driver framework (UMDF) drivers for other device types must rely on the kernel-mode device stack to manage idle detection, power-down on idle, and wake.
* Windows Driver Model (WDM) drivers can either rely on the Windows power manager to perform idle detection and issue device power state (D*x*) I/O request packets (IRPs) to change the device power state or can implement their own idle-detection code that sends D*x* IRPs.

For details, see “Increasing System Power Efficiency through Driver Support for Runtime Idle Detection” on the WHDC Web site and “Detecting an Idle Device” in the WDK documentation. [*[source]*](#_Device_and_Driver)

## Integration with Windows Power Policy

All drivers should integrate device power policy with the Windows power policy. If the device supports one or more device performance states—that is, a lower power/lower performance mode—the driver should integrate the policy for entering those states with system power policy:

* Drivers should enable the most conservative power savings by default when on DC power.
* If possible, drivers should implement adaptive behaviors and should enable them by default for the Balanced power scheme for both AC and DC power.

Devices that implement device performance states might require additional custom driver support, as follows:

* Drivers that change device power behavior to correspond to system power settings should register for notification of changes in the system’s power settings.
* Drivers can define custom power policy settings to support device-specific behaviors.

### Power Setting Notification

Drivers register to be notified when a system power setting changes by calling a system notification function and supplying the globally unique identifier (GUID) of the setting for which notification is required. The WDK defines GUIDs for numerous device-type specific settings, along with the following system-wide settings:

* Power source (GUID\_ACDC\_POWER\_SOURCE)
* Power scheme personality (GUID\_POWERSCHEME\_PERSONALITY)

Personalities provide a way to indicate the intention of a power scheme. Each system-defined power scheme is associated with a personality, and a personality is also associated with a custom power scheme.

To register for power setting notification in a kernel-mode (KMDF or WDM) driver

1. Implement one or more power notification callback functions that perform device-specific actions that are required when the system power setting changes.

2. Call PoRegisterPowerSettingCallback from the DriverEntry or AddDevice function to associate a callback function with a particular power setting.

When the setting changes, the Windows power manager invokes the callback and passes the setting and its new value.

For more information about **PoRegisterPowerSettingCallback**, see “PoRegisterPowerSettingCallback” in the WDK documentation. [*[source]*](#_Device_and_Driver)

To register for power setting notification in a UMDF driver

1. Register for notification of power policy changes by calling RegisterPowerSettingNotification with the GUID of the setting.

2. Handle the received WM\_POWER\_BROADCAST message when the power setting changes.

### Custom Power Policy Settings

If your device supports several performance or power-saving states, you might want to create a custom power scheme that integrates these states.

You can create a custom power setting either in the INF or in a user-mode installation application.

To create a custom power setting in the INF

* Use the AddPowerSetting directive to define a power setting, its possible values, and its default in each power scheme personality. See “INF AddPowerSetting Directive in MSDN” for details and an example. [*[source]*](#_Device_and_Driver)

To create and install a custom power scheme in an application

* Use the Windows Power API as described in “Power Scheme Management” on the MSDN® Web site. [*[source]*](#_Device_and_Driver)

You can also change power schemes by using the PowerCfg command-line tool.

When you create a custom power scheme, the system displays it in the Control Panel Power application along with the system-defined schemes. User-mode power configuration applications are not required.

# User-Mode Services and Device-Specific Applications

Many drivers come with user-mode services or applications. These components require system resources and can affect system power consumption and battery life in the same manner as any other type of application does.

##### Recommendations for Services and Applications Developers

Device-specific services and applications should conform to best practices for power-managed software:

* Invest in performance optimizations.

Important: Every performance optimization is a battery life optimization. Reductions in required resources, such as using less processor time, allow system hardware to become idle and enter low-power modes.

* Adjust to user power policy by reducing resource usage or increasing performance accordingly.
* Reduce the frequency of periodic background tasks such as updates when the system is on battery power.
* Avoid polling and spinning in tight loops, which use the processor heavily and reduce the effectiveness of processor idle states and processor performance states.

Use interrupts, events, and callback functions instead.

* Do not use high-resolution (<15‑ms) periodic timers, which similarly reduce the efficiency of processor power-management technologies.

If a high-resolution timer is absolutely required, ensure that the application disables the timer as soon as the task is complete.

* Respond to common power events, such as power plan personality and system power source (AC/DC) changes, to help extend system battery life.

For additional details, see “Application Power Management Best Practices for Windows Vista” on the WHDC Web site. [*[source]*](#_Device_and_Driver)

Test all device-specific applications to ensure that they follow the best practices for power management. Microsoft supplies the following test tools:

* The PwrTest utility is supplied with the WDK.
* The Power Event Monitoring Tool is provided as a sample Microsoft® Visual Studio® 2005 project that developers can compile and build. The tool sample requires the Windows Vista Software Development Kit (SDK). [*[source]*](#_Device_and_Driver)

# Device Class–Specific Information

This section provides information and recommendations about selecting devices and implementing drivers for the following device classes:

PCI Express

Graphics

USB Hubs and Controllers

Wireless Devices

Storage Devices

Audio Devices

## PCI Express

PCI Express devices must support Active State Power Management (ASPM). ASPM allows the system to incrementally reduce power to serial links in a PCI Express (PCIe) fabric as the link becomes less active.

Windows Vista enables or disables ASPM for a link based on the overall system power policy, the hardware capabilities of the link, and the latency of the link. Windows Vista configures the PCIe fabric to enable the maximum power savings.

Because the majority of devices that comply only with PCI Express Base Specification 1.0 did not implement ASPM correctly, Microsoft selected PCI Express Base Specification 1.1 as the required level of support to enable ASPM. Thus, some devices must opt out of ASPM because of compatibility problems. For more information, see “Active State Power Management in Windows Vista” on the WHDC Web site. [*[source]*](#_Device_and_Driver)

##### Recommendation for System Manufacturers

* Validate that each PCIe device supports and uses ASPM.

To manage ASPM support at system level

1. Windows Vista determines PCI Express Base Specification revision compliance by using the Role-based Error Reporting capability bit in the Device Capabilities register.

This bit was one of the last revisions adopted before the release of Revision 1.1 of the *PCI Express Base Specification*. If a device or endpoint does not support this bit in the Device Capabilities register, Windows assumes that it is not compatible with Revision 1.1 or later and thus does not enable ASPM.

2. The system BIOS can enable ASPM on pre-1.1 devices and can disable ASPM systemwide.

Windows can override these settings for a particular device depending on the capabilities and settings of the device itself.

To manage ASPM support at device level

1. Report ASPM capabilities in the Link Capabilities register.

Enable L0s and L1 Link capabilities only when devices truly support and have been tested with ASPM.

2. Report accurate L1 exit latency values in the Link Capabilities register to allow Windows Vista to calculate exit latencies as precisely as possible.

3. To enable ASPM support on known good hardware that predates *PCI Express Base Specification*, Revision 1.1, include the following in the device INF file:

[DDInstall.HW]
Include=machine.inf
Needs=PciASPMOptIn

For information about setting ASPM-related power policies, see “[Power Policy Recommendations for Extended Battery Life](#_Power_Policy_Recommendations)” in Part 3. For additional details about ASPM on Windows Vista and later versions, see “Active State Power Management in Windows Vista” on the WHDC Web site. [*[source]*](#_Device_and_Driver)

## Graphics

Graphics devices should integrate device performance states with Windows power policy. Device vendors should define device performance states that scale appropriately for the power source (AC or battery). For example, a graphics device might reduce power consumption by implementing one or more of the following:

* Scalable GPU core voltage and frequency
* Scalable memory clock
* Reductions in refresh rate

Drivers define and install power settings and register for notification of power setting changes as described in “[Integration with Windows Power Policy](#_Integration_with_Windows)” earlier in this part of the guide.

Windows Vista integrates display brightness into power policy, so that users can easily control the brightness of the display. Each power scheme is associated with a brightness-level preference with different values for AC and DC power sources.

### VSync Interrupt

The Desktop Window Manager (DWM) uses VSync interrupt (also known as vertical blank interrupt or VBI) to synchronize display updates to the video adapter. The GPU generates a VSync interrupt at the monitor’s vertical refresh rate, and the processor must wake up to service it.

VSync can increase system power consumption because it:

* Reduces the efficiency of processor idle states.
* Requires clocks within the GPU to remain enabled so that they can generate the interrupt.
* Prevents some GPU power-management technologies.

Overall, Microsoft estimates that VSync increases system power consumption by 1 to 2 percent.

In Windows Vista SP1, VSync behavior has been changed to conserve power. VSync no longer runs continuously. Instead, it is disabled after a short timeout if no screen updates have occurred, such as when displaying a static desktop.

To take advantage of this capability in Windows Vista SP1 and later releases, both the graphics device and driver must support disabling VSync, as described in “VSync Control in Windows Vista SP1” on the WHDC Web site. [*[source]*](#_Device_and_Driver)

### Graphics and Display Driver Considerations

##### Recommendations for Graphics and Display Drivers

* Support brightness control through the ACPI brightness object.

For implementation details, see “Supporting Brightness Controls on Integrated Display Panels” in the WDK documentation. [*[source]*](#_Device_and_Driver)

* Support disabling VSync for Windows Vista SP1 and later releases.
* Do not wait in a busy loop in the display driver, which maintains high CPU usage.

## USB Hubs and Controllers

##### Recommendations for System Manufacturers

Although as a system vendor you cannot control the number or types of USB devices that end users attach, you should ensure that the USB support in your mobile platforms conforms to two important guidelines:

* Support selective suspend in all USB devices.
* Avoid integrating USB hubs into mobile platforms.

All of the USB devices that are attached to the PC must support selective suspend for the USB host controller to enter its global suspend mode. The USB host controller must be in global suspend before the CPU can enter deep processor idle (C) states, including C3 and C4 on some processors.

### Integrated USB Hubs

Whenever possible, avoid integrating nonroot USB hubs into mobile platforms for several reasons:

* Hubs require additional power.
* Hubs increase the DMA activity of the host controller to which they are attached, which results in less efficient processor power management.
* Hubs increase the time that is required for the system to resume from S3.

If a particular platform must include a hub, make sure that any input devices are attached directly to a host controller root hub so that they are immediately available when the system resumes.

Avoid using USB devices that incorporate additional hubs. Such hubs can limit the ability of the device to enter sleep states and can thus increase power consumption. In multifunction devices, ensure that the device is a true composite device that has different endpoints for each function, and not a device that has multiple functions behind an integrated USB hub.

### USB Host Controllers

If your platform incorporates a USB 2.0 host controller, you must ensure that the capabilities of the controller are properly reported. The ACPI driver and the USB host controller driver use the reported capabilities to implement system power-management strategies.

For detailed information about reporting these capabilities and about how the ACPI driver and USB host controller driver use them, see “Power Management of USB Host Controllers” on the WHDC Web site. [*[source]*](#_Device_and_Driver)

### USB Selective Suspend

Every USB device must support selective suspend. Selective suspend allows an idle device to be put into a suspended state without affecting other devices that are connected to the same hub or—in the case of a multifunction device—without affecting the other functions in the device. After all of the devices or functions have been suspended, the entire hub or multifunction device can be powered down.

How a driver supports and enables USB selective suspend depends upon the underlying driver model: UMDF, KMDF, or WDM. Consider these guidelines:

* Implement USB drivers by using UMDF if possible.

UMDF relies on the underlying Microsoft-supplied WinUSB mechanism to support selective suspend. Very little work is required on the part of the driver writer.

* If the device requires features or resources that are available only in kernel mode, use KMDF.
* Modify existing WDM drivers to support runtime idle detection and selective suspend.

However, properly implementing selective suspend in a WDM driver is quite difficult. It requires significant changes to existing code and relatively complicated new code, and no samples exist. We recommend that such drivers instead be converted to KMDF (or UMDF, if possible). For a guide to driver conversion, see “Porting a Driver from WDM to KMDF” on the WHDC Web site. [*[source]*](#_Device_and_Driver)

Important: All new USB drivers should be Windows Driver Foundation (WDF) drivers.

#### Selective Suspend in UMDF Drivers

UMDF drivers rely on the selective suspend features that are built into the Microsoft-supplied WinUSB.sys kernel-mode driver. A UMDF USB driver must notify WinUSB.sys that the driver and device support selective suspend and must enable selective suspend and supply an idle timeout value. WinUSB.sys then manages idle detection and device suspension on behalf of the UMDF driver. WinUSB.sys must be the power policy owner for the device stack.

To enable USB selective suspend in a UMDF driver

* To notify WinUSB.sys that the device can support USB selective suspend, the AddReg directive in the device INF includes the DeviceIdleEnabled value for the device’s hardware key and sets the value to 1. For example:

HKR,,"DeviceIdleEnabled",0x00010001,1

A UMDF driver can enable USB selective suspend support in WinUsb.sys either during installation or at runtime:

* To enable support during installation, the AddReg directive in the device INF adds the DefaultIdleState value to the device’s hardware key and sets the value to 1, as the following shows:

HKR,,"DefaultIdleState",0x00010001,1

* To enable support at runtime, the function driver calls IWDFUsbTargetDevice::SetPowerPolicy, with the method’s PolicyType parameter set to AUTO\_SUSPEND and the Value parameter set to 1.

By default, WinUSB suspends the device when no transfers are pending. To indicate that WinUSB should wait after a transfer before it suspends the device, the UMDF driver must specify a timeout value.

To set a timeout value in a UMDF driver

* Call IWDFUsbTargetDevice::SetPowerPolicy with PolicyType set to SUSPEND\_DELAY and Value to the idle timeout value, in milliseconds.

A UMDF USB function driver can optionally allow the user to enable or disable selective suspend.

To enable user control of USB selective suspend in a UMDF driver

* Include an AddReg directive in the INF that adds the UserSetDeviceIdleEnabled value to the device’s hardware key and sets the value to 1:

HKR,,"UserSetDeviceIdleEnabled",0x00010001,1

If UserSetDeviceIdleEnabled is set, the device’s Properties dialog box includes a Power Management tab that allows the user to enable or disable USB selective suspend.

#### Selective Suspend in KMDF Drivers

If the USB device requires features or resources that are not available in user mode, you should supply a KMDF driver. KMDF drivers implement selective suspend by setting relevant values in a KMDF initialization structure and then supplying the appropriate callback functions.

To implement selective suspend in a KMDF function driver

1. In the EvtDriverDeviceAdd callback, initialize a WDF\_DEVICE\_POWER\_POLICY\_
IDLE\_SETTINGS structure and use it to specify idle settings, including the following:

* Whether the device supports wake when the system is in S0.
* Whether the device supports USB selective suspend.
* Which Dx state the device enters when it is powered down.
* The idle timeout value.
* Whether the user can modify the device’s idle settings.
* Whether the device supports runtime idle detection.

2. Pass the structure to WdfDeviceAssignS0IdleSettings to specify the idle policy.

3. Implement EvtDeviceD0Entry and EvtDeviceD0Exit to handle device power up and power down, respectively.

Note: If a function driver uses a continuous reader on a USB endpoint, the reader’s repeated polling does not count as activity toward the KMDF idle timer. If no data is actually read, KMDF considers the device idle and powers it down when the idle timer expires.

For a detailed walkthrough of sample code that implements selective suspend and device wake in a KMDF driver for USB, see “Plug and Play and Power Management in WDF Drivers” on the WHDC Web site. [*[source]*](#_Device_and_Driver)

#### Selective Suspend in WDM Drivers

To support USB selective suspend, a WDM driver requires fairly complicated code. The following is a broad outline of the minimal requirements for supporting selective suspend:

* The driver must implement an idle timer to determine when to suspend the device.

USB client drivers should use the USB selective suspend request rather than PoRegisterDeviceForIdleDetection.

* When the idle timer expires, the function driver sends a USB selective suspend request to the parent driver that notifies it to power down the device.

The USB selective suspend request is the internal device I/O control request with the IOCTL\_INTERNAL\_USB\_SUBMIT\_IDLE\_NOTIFICATION control code. The request includes a pointer to the function driver’s USB selective suspend callback function.

* The parent driver initiates the power-down process as soon as it is appropriate to do so.
* The function driver does not start its power-down operation until its selective suspend callback function is invoked.

For details on how to add this support to a WDM driver, see “USB Selective Suspend” in the WDK documentation. See also “Increasing System Power Efficiency through Driver Support for Runtime Idle Detection” on the WHDC Web site. [*[source]*](#_Device_and_Driver)

The USB selective suspend callback function should synchronously power down the device, as follows:

1. If the device can wake itself, the callback arms the device for wake from S0.

2. The callback requests a Dx IRP and sends it to the top of the stack.

3. When the parent driver receives the Dx IRP, it powers down the device.

4. After the device has transitioned to Dx, the USB selective suspend callback function returns.

5. After the device returns to D0, the parent driver completes the USB selective suspend request as successful.

The I/O manager calls the WDM function driver’s I/O completion routine to notify the driver that the USB selective suspend operation is finished. If the device is surprise removed while it is powered down, the parent driver completes the USB selective suspend request as canceled.

Function drivers usually are not required to synchronize USB selective suspend request completion with the device’s power policy state.

Note: For USB devices, D1 through D3 are software constructs and have essentially the same wake latency. For Windows XP, D2 is the level that can be armed for wake, so function drivers for such devices should transition the device to D2. For Windows Vista, a device in D2 or D3 can be armed for wake.

## Wireless Devices

Mobile platforms support more types of wireless devices than ever before, and the number and types of wireless devices in individual configurations continues to increase. Such devices include:

* 3G technologies
* 802.11*x* networking
* Bluetooth wireless technology
* Ultra-Wideband (UWB)
* Worldwide Interoperability for Microwave Access (WiMax)

As with every other device type, wireless devices use power for their own operation and can affect the power usage of other devices in their device stack, such as USB hubs and ports. Even wireless devices that are infrequently used can affect battery life if they cannot be powered off when they are idle.

### Wireless Device Selection

All wireless devices must be able to power down when not in use.

Select wireless devices carefully. Integrate only devices that are required for the intended market; do not build in extraneous hardware that your customers won’t use. Additional devices consume additional power. Specific guidelines include the following:

* Use Bluetooth devices that support sniff subrating, as defined in Bluetooth Core Specification Version 2.1 + Enhanced Data Rate (EDR). [*[source]*](#_Device_and_Driver)

Sniff subrating decreases the frequency with which the device “sniffs” for activity in low-power modes and has been shown to have marked impact on mobile PC battery life.

* Use integrated radios—such as UWB and Bluetooth in the same module—whenever feasible.

Such devices share components and thus avoid duplicating power-consuming equipment. For UWB and Bluetooth, in particular, the Bluetooth component can find the communications partner and manage the device while the UWB component handles tasks that require its high data-transmission rates.

### Wireless Driver Considerations

Drivers for wireless devices should integrate their power policy with Windows power policy. Specific details depend upon the type of device and the required driver model. For example:

* Network Driver Interface Specification (NDIS) miniport drivers for 802.11 devices must supply a MiniportPnpEventNotify callback that handles the power-profile changed events.

For details, see “802.11 Miniport Driver Power Management” in the WDK documentation. [*[source]*](#_Device_and_Driver)

* Bluetooth Logical Link Control and Adaptation Protocol (L2CAP) drivers should support power management as defined for the underlying device type connection, such as USB, and should integrate with Windows power policy according to the driver model that is required for that device.
* Bluetooth radio device drivers should use UMDF and should integrate with Windows power policy through the UMDF model.
* Infrared (IR) device drivers should use KMDF and integrate with Windows power policy through the KMDF model.
* Bluetooth radios must support selective suspend. If a radio scan is required, the driver should poll periodically and enter suspend between polling periods.

##### Recommendations for System Manufacturers

At a minimum:

* Wireless radios should automatically power off when they are not in use.
* Wireless devices should support appropriate power settings for each of the Windows power profiles.

## Storage Devices

Careful selection and configuration of storage devices are critical to building mobile platforms with long battery life. All disks should support idle detection and low-power states. This section provides guidelines for selecting and configuring certain types of storage devices and guidelines for disk-related power settings.

### Serial ATA Devices with AHCI Chipsets

Serial ATA (SATA) devices on systems that support the Intel *SATA Advanced Host Controller Interface (AHCI) Specification* must support and be tested for both host-initiated power management (HIPM) and device-initiated power management (DIPM). Vendors must ensure that the operating system support for these features is enabled in the registry, as described in [Part 3](#Part3) of this guide. [*[source]*](#_Device_and_Driver)

##### Recommendation for System Manufacturers

* AHCI optical drives should support asynchronous notification (AN), which helps reduce the need for the operating system to poll optical drives for disc insert and ejection status.

### USB-Connected Storage Devices

Integrated USB-connected storage devices should be avoided in most circumstances because the native Windows USB storage driver does not support selective suspend.

##### Recommendation for System Manufacturers

* If a USB-connected storage device is integrated into the system, provide additional software and hardware to automatically disconnect and power off the device when it is not in use.

### Disk Idle Detection

All disks should support idle detection and low-power states. Disk idle detection is key to achieving maximum mobile PC battery life because the difference in power consumption between the spin-up and spin-down states may be 500 megawatts (mW) or more on mobile PC form-factor disk devices.

The Windows power manager uses disk activity, power policy, and system power source (AC/DC) to determine when to spin down and spin up a disk. Power policy configures the time interval between the last disk access and spin-down of the device as well as any amount of disk activity that should be ignored to be more aggressive in entering the spin-down state.

##### Recommendations for Device and System Manufacturers

* Customize disk idle and burst ignore timeouts for hardware.
* Set appropriate disk idle detection timeout values and disk burst ignore threshold values in system policies.

For maximum power savings on mobile platforms, reduce the disk idle detection timeout from the default value of 30 or 60 minutes. To further favor power savings, set the disk burst ignore threshold to a non-zero value between 15 and 60 seconds. In determining timeout values, consider that:

* Minimum idle timeout is 30 seconds.
* Spinning down the disk for less than 20 seconds uses more energy than keeping the drive spun up.

Evaluate the disk idle detection policies on your specific configurations with value-added software installed. Periodic writes and indirect operations such as file creates, file extends, and registry writes can prevent the disk from spinning down.

PerfMon supports counters that can be useful in measuring disk idle times and burst intervals. For details, see “Disk Subsystem Performance Analysis for Windows” on the WHDC Web site. [*[source]*](#_Device_and_Driver)

To configure disk idle detection policies

* Use Powercfg to configure disk idle detection policies.

For details, see “Windows Disk Idle Detection” on the WHDC Web site. See also “PowerCfg Command-Line Options” on the TechNet Web site. [*[source]*](#_Device_and_Driver)

To observe disk idle detection

* Use the PwrTest /DISK scenario to observe disk idle detection.

For details, see “Windows Disk Idle Detection” on the WHDC Web site. See also “PwrTest” in the WDK documentation. [*[source]*](#_Device_and_Driver)

## Audio Devices

Intel High Definition Audio (HD Audio) is the most common integrated audio solution. The *HD Audio Specification*, which is available on the Intel Web site, defines how codecs can power down after a timeout when inactive. To help reduce power consumption, all HD Audio devices and drivers should support this feature. [*[source]*](#_Device_and_Driver)

##### Recommendation for System Manufacturers

* Integrate your audio adapter and miniport drivers with the Microsoft audio port class driver (PortCls.sys).

The PortCls.sys driver implements most of the generic functionality that HD Audio devices require and handles power requests on behalf of the audio driver. In Windows Vista, the PortCls driver uses the following defaults:

* Conservation idle timeout is 0 (never idle).
* Performance idle timeout is 0 (never idle).
* Power state for an idle device is D0.

Consequently, the device never idles out and thus remains in the working state.

In Windows Vista SP1, these defaults change as follows:

* Conservation idle timeout is 30 seconds.
* Performance idle timeout is 0 seconds (never idle).
* Codecs integrated with PortCls enter D3 when idle on battery power.

If the audio device can change power states in response to system requests:

* The adapter driver should expose the IAdapterPowerManagement interface on the adapter object.
* The miniport driver should expose the IPowerNotify interface on any miniport object or stream object that requires notification of impending power-state changes.

For details, see “Power Management for Audio Devices” in the WDK documentation. [*[source]*](#_Device_and_Driver)

### Optimization of Audio Hardware Power Consumption

##### Recommendations for System Manufacturers

Consider the following guidelines for optimizing audio hardware power consumption:

* Select and integrate audio devices (codecs) that can enter D3 at runtime without audible pops or clicks on entering D3 or returning to D0.
* Use audio codecs that support D3 when all device paths are idle.
* Support Sony/Philips Digital InterFace (S/PDIF) keep-alive during D3 if S/PDIF is the default audio device.
* Use audio devices that support granular power changes to put unused audio device paths (all widgets associated) into D3 when these are idle.
* Remove the 8254 timer chip so that the system can play accessibility beeps when the audio device is powered off.

If accessibility beeps from the 8254 timer chip are required, implement a setting in the Control Panel Audio Device application to allow the timer chip to be powered down. This allows users who do not need accessibility sounds to reduce power consumption.

* Remove the analog CD connection from the CD-ROM/DVD-ROM to the audio device to avoid requiring the audio device to be powered on indefinitely so that the user can listen to audio CDs.
* If possible, avoid polling of S/PDIF input.

However, if the audio device subsystem requires such polling:

* Design the driver to poll at 1- to 2‑second intervals and keep the codec in D3 during the interval between the polls to detect attached devices.
* Use a codec that either implements “Lock” detection hardware for the polling process or that can discover and report a digital signal on S/PDIF input without polling.
* Use Class D amplifiers for speakers and power them down when speakers are not being used (device is idle).
* Use High-Definition Multimedia Interface (HDMI) audio hardware that can put the display into a low-power mode without stopping audio transmission when the video is not being used.
* Enable proprietary hardware encryption only when absolutely necessary.

### Optimization of Audio Software and Driver Power Consumption

##### Recommendations for System Manufacturers

Consider the following guidelines for optimizing power consumption by audio software and drivers:

* Use Windows Vista SP1 to enable key power-management enhancements for audio devices.
* Use audio drivers that take advantage of the Windows audio system’s PortCls idle timer functionality (SP1 and later only).

This functionality ensures that the driver sets hardware to a low-power state after a configurable idle period (system is not rendering audio data for a period of time).

The native HD Audio class driver uses this capability, whereas some third-party audio drivers do not. Ensure that the driver for your audio subsystem takes advantage of the PortCls idle timer functionality.

* Use drivers that monitor the power plan that the user selects.

Also:

* Instantiate lower complexity audio processing algorithms in the audio pipeline when on battery.
* Provide reduced-capability stream rates/widths—for example, reduce the bit rate to 48 kilohertz (kHz)—when on battery power. Reduced capabilities lower processing demands, so that the CPU can enter lower power states more often.

# Devices and Drivers Recommendations

##### Recommendations for System Manufacturers

Systemwide:

* Ensure that all devices implement runtime idle power management.

For wireless devices:

* Ensure that the user can disable wireless devices to save power.

For PCIe devices:

* Ensure that all PCIe devices support ASPM.

For graphics devices:

* Incorporate graphics device hardware that can automatically reduce power consumption when the system is on battery power.
* Validate that device power controls and power states are integrated with Windows power policy.

For USB devices:

* Validate that all integrated USB devices support selective suspend and automatically enter the suspend state when idle.
* Avoid integrating USB hub devices. USB hub devices draw extra battery power and extend resume performance.
* Validate selective suspend support for USB-connected Bluetooth radios.
* Avoid integrating USB devices that do not support selective suspend, including mass-storage devices such as optical drives and USB flash drives.

For storage devices:

* Validate that SATA storage devices support both HIPM and DIPM.
* Enable DIPM in the software preinstallation when using the Microsoft SATA storage driver (Msahci.sys).

For audio devices:

* Incorporate audio codec hardware that supports runtime idle power management by supporting the D3 power state.

For integrated platform devices:

* Implement sensor devices such as accelerometers, sensors, and buttons by using an interrupt model or ACPI notifications to avoid software polling to detect trip and failure conditions.

# Device and Driver Selection References

White papers on the WHDC Web site:

Active State Power Management in Windows Vista
<http://www.microsoft.com/whdc/system/bus/PCI/aspm.mspx>

Application Power Management Best Practices for Windows Vista
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PM_apps.mspx>

Disk Subsystem Performance Analysis for Windows
<http://www.microsoft.com/whdc/device/storage/subsys_perf.mspx>

Increasing System Power Efficiency through Driver Support for Runtime Idle Detection
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/s0idle_driver.mspx>

Plug and Play and Power Management in WDF Drivers
<http://www.microsoft.com/whdc/driver/wdf/WDF_pnpPower.mspx>

Porting a Driver from WDM to KMDF
<http://www.microsoft.com/whdc/driver/wdf/WDF_Port.mspx>

Power Management of USB Host Controllers
<http://www.microsoft.com/whdc/system/bus/usb/USB-HC_PwrMgmt.mspx>

VSync Control in Windows Vista SP1
[http://www.microsoft.com/whdc/device/display/wddm\_vsync.mspx#](http://www.microsoft.com/whdc/device/display/wddm_vsync.mspx)

Windows Disk Idle Detection
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/Disk_Idle_Detection.mspx>

Specifications:

Bluetooth Core Specification Version 2.1 + Enhanced Data Rate (EDR)
<http://bluetooth.com/Bluetooth/Technology/Works/Core_Specification_v21__EDR.htm>

Intel High Definition Audio Specification
<http://www.intel.com/standards/hdaudio/>

*Intel SATA Advanced Host Controller Interface (AHCI) Specification* (Intel Web site)
<http://download.intel.com/technology/serialata/pdf/rev1_2.pdf>

PCI Express Base Specification 1.1
<http://www.pcisig.com/specifications/pciexpress>

WDK Documentation and Other Topics on MSDN:

802.11 Miniport Driver Power Management
<http://msdn2.microsoft.com/en-us/library/aa504155.aspx>

Detecting an Idle Device
<http://msdn2.microsoft.com/en-us/library/ms798268.aspx>

INF AddPowerSetting Directive
<http://msdn2.microsoft.com/en-us/library/aa906359.aspx>

PoRegisterPowerSettingCallback
<http://msdn2.microsoft.com/en-us/library/aa492389.aspx>

Power Management for Audio Devices
<http://msdn2.microsoft.com/en-us/library/ms790569.aspx>

Power Scheme Management (in the “Win32 and COM Development” library)
<http://msdn2.microsoft.com/en-us/library/aa372711.aspx>

Supporting Brightness Controls on Integrated Display Panels
<http://msdn2.microsoft.com/en-us/library/aa906355.aspx>

USB Selective Suspend
<http://msdn2.microsoft.com/en-us/library/ms793200.aspx>

Windows Vista SDK
<http://www.microsoft.com/downloads/details.aspx?familyid=4377f86d-c913-4b5c-b87e-ef72e5b4e065&displaylang=en>

Tools:

PowerCfg Command-Line Options
<http://technet2.microsoft.com/WindowsVista/en/library/1d58b934-f56a-4796-b2df-7be2eb9c03bc1033.mspx?mfr=true>

PwrTest
<http://msdn2.microsoft.com/en-us/library/aa906552.aspx>

PART 3:
Operating System Configuration of Power Policy

In this part

Introduction to Power Policy Configuration

Power Policy Infrastructure

Power Plans

Power Policy User Experience Elements

Optimizing Power Policy for Extended Battery Life

Display Idle Timeout

Display Brightness

Hard Disk Drive Idle Timeout

802.11 Wireless Power-Save Mode

Idle Detection Threshold

SATA AHCI Link Power Management

Interlaced DVD Playback

Optimizing Processor Power Management Policy

Favoring Power over Performance

Reducing Maximum Performance State

Power Policy Recommendations for Extended Battery Life

Power Policy Optimization References

# Introduction to Power Policy Configuration

Power policy refers to Windows infrastructure that helps system manufacturers configure their hardware platform and devices for power savings and extended battery life. After hardware platform and device components have been selected for a system, power policy can be used to tailor that system to favor power savings or performance.

Power policy includes basic system power-management settings such as the display and sleep idle timeouts. Advanced configuration options are provided for processor power management, USB selective suspend, and wireless network adapter power modes. Power policy can also contain configuration options for devices and applications that are added to the system, including vendor-specific power settings for the graphics device.

Power policy settings can be configured by using the PowerCfg utility in Windows, which provides command-line access to power policy settings, power plans, and other power-management information. PowerCfg can be used to customize settings within a given power plan or to create entirely new power plans for the system.

Some power policy settings and their actions can be validated by using the PwrTest utility, which is available in the WDK. PwrTest allows an administrator to validate correct operation of the device and disk idle timeout, as well as processor power-management behavior. For more information, see “PwrTest” in the WDK documentation. [*[source]*](#_Power_Policy_Optimization)

We encourage system manufacturers to review the basic and advanced power policy settings and then customize settings for each specific mobile platform to help extend battery life. These settings can be embedded in a preinstallation image for a particular platform. This part of the guide recommends specific configuration settings that favor reduced power savings over performance to help extend mobile PC battery life.

# Power Policy Infrastructure

The power policy store in Windows Vista was redesigned to enable new system power-management features, including third-party extensibility and deployment of power policy settings by using Windows Group Policy. The power policy store contains definitions for power settings and power plans that allow the end user to change multiple power policy values with a single profile.

Each power setting configures a single power-management behavior. Examples of power settings include the Display Idle Timeout and the System Power Button action.

The following example uses the PowerCfg command-line tool to display the definition of the Display Idle Timeout power setting, which configures the period of inactivity before the system display is automatically powered off:

C:\>powercfg /q SCHEME\_CURRENT SUB\_VIDEO

Power Scheme GUID: 381b4222-f694-41f0-9685-ff5bb260df2e (Balanced)

 Subgroup GUID: 7516b95f-f776-4464-8c53-06167f40cc99 (Display)

 Power Setting GUID: 3c0bc021-c8a8-4e07-a973-6b14cbcb2b7e

 (Turn off display after)

 Minimum Possible Setting: 0x00000000

 Maximum Possible Setting: 0xffffffff

 Possible Settings increment: 0x00000001

 Possible Settings units: Seconds

 Current AC Power Setting Index: 0x0000012c

 Current DC Power Setting Index: 0x0000012c

PowerCfg defines aliases for common power-related GUIDs. In the preceding example, SCHEME\_CURRENT is an alias that identifies the current active power plan, instead of specifying the full GUID of the active power plan. Similarly, SUB\_VIDEO is an alias for the GUID of the video settings subgroup. For more examples of using PowerCfg, see the “Power Policy Configuration and Deployment in Windows Vista” paper on the WHDC Web site and “PowerCfg Command-Line Options” on the TechNet Web site. [*[source]*](#_Power_Policy_Optimization)

## Power Plans

A power plan—sometimes called a power scheme—contains the power setting values that are in effect on a system. Only one power plan can be active at a time. The end user changes the system’s overall power savings or performance characteristics by changing the system’s power plan.

A power plan contains the following:

* Power plan GUID
This GUID uniquely identifies the power plan to distinguish it from all other power plans on the system.

Software and configuration utilities can refer to the power plan by using this unique identifier instead of a zero-based integer index or name string, as was required by earlier versions of Windows.

* Power plan friendly name and description strings
The friendly name and description identify the power plan in the Control Panel Power Options application and in the PowerCfg command-line tool.
* Power plan personality attribute
This attribute describes the overall power-saving behavior of the plan.

The personality attribute helps indicate to software what the user intends for power savings or performance.

* AC and DC value preferences for power settings
A power plan contains an AC and a DC value preference for each power setting:
* The AC value is used when the system is on AC power (also called utility power). AC values for battery settings such as the critical battery threshold are not used on any system type because such settings don’t apply when the system is on AC power.
* The DC value is used when the system is on battery power. DC values are used only on systems that have a battery or an attached uninterruptible power supply (UPS).

Note: The power setting definition lists the potential values for a power setting. However, the power plan contains the actual value preference that the system is to use for that power setting. This enables the user to change preferences for multiple power settings at once simply by changing power plans.

More than one power plan can be installed on a system, although only one is in effect at any particular time. Table 3‑1 lists the default power plans that are installed with Windows Vista.

Table 3-1 Default Windows Vista Power Plans

| Friendly name | Description | GUID |
| --- | --- | --- |
| Automatic/ Balanced | Automatically balances performance with energy consumption on capable hardware. (Default power policy.) | 381b4222-f694-41f0-9685-ff5bb260df2e |
| High Performance | Favors performance, but might use more energy. | 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c |
| Power Saver | Saves energy by reducing the computer’s performance where possible. | a1841308-3541-4fab-bc81-f71556f20b4a |

The following example uses PowerCfg to enumerate the power plans that are installed on a system:

C:\>powercfg /list

Existing Power Schemes (\* Active)

-----------------------------------

Power Scheme GUID: 381b4222-f694-41f0-9685-ff5bb260df2e (Balanced) \*

Power Scheme GUID: 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c (High performance)

Power Scheme GUID: a1841308-3541-4fab-bc81-f71556f20b4a (Power saver)

## Power Policy User Experience Elements

The end user can easily control energy savings and adjust power policy configuration on a system by using the new Windows battery meter and the Control Panel Power Options application, shown in Figure 3‑1.

By using the Control Panel Power Options application, the user can:

* View the list of installed power plans.
* Change individual power setting values within a specified power plan, such as the Display Idle Timeout power setting.
* Create new power plans or restore the default settings for a power plan.



Figure 3‑1 Control Panel Power Options Application

The Advanced settings tab (shown in Figure 3‑2) in the Control Panel Power Options application allows the user to view and change all power settings that are installed on the system. Advanced settings includes power settings that have been installed by third-party application and device software.



Figure 3-2 Power Options Advanced Settings

# Optimizing Power Policy for Extended Battery Life

System manufacturers can configure Windows-based platforms for maximum energy efficiency by manipulating power policy and other system configuration settings. The default Windows Vista power policy is designed to support a wide breadth of hardware platforms and to balance power savings and energy efficiency with performance, but the policy can be changed to favor performance or power savings to help extend mobile PC battery life.

The PowerCfg utility can be used in cooperation with preinstallation image utilities and command-line script utilities to make the recommended power policy. System manufacturers should validate each power policy change on their platforms to ensure reliable operation with selected platform hardware and devices.

System manufacturers can customize the power policy settings described in this section to help extend mobile PC battery life.

## Display Idle Timeout

On a modern mobile PC, the attached display is the largest power consumer, requiring up to 40 percent of the overall system power-consumption budget. The display idle timeout allows Windows to power down the attached display after a period of user inactivity. The display consumes very little power when it is off. Usable mobile PC battery life can be extended by a significant amount by using a short display idle timeout.

By default, Windows Vista configures the display idle timeout to a moderately short value for mobile and desktop systems. Table 3‑2 list details for configuring the display idle timeout.

Table 3-2 Details for Display Idle

| **Detail** | **Description** |
| --- | --- |
| Friendly name  | Turn off the display |
| Description  | Specifies how long the computer is inactive before the display turns off.  |
| PowerCfg alias\*  | VIDEOIDLE  |
| Group policy path | Administrative Templates\System\Power Management\Video and Display Settings\Turn Off the Display After |
| GUID  | 3c0bc021-c8a8-4e07-a973-6b14cbcb2b7e  |
| Defined in | Ntpoapi.h |
| Balanced defaults  | 20 minutes (AC)  | 5 minutes (DC)  |
| \*The **PowerCfg** alias can be used in place of the GUID in various commands. For more information, see “Power Policy Configuration and Deployment in Windows Vista” on the WHDC Web site. [*[source]*](#_Power_Policy_Optimization) |

## Display Brightness

The most effective way to reduce the power consumption of the attached display in a mobile PC when the display is in use is to reduce its brightness. In LCD displays, the backlight is responsible for the majority of the display’s power consumption. Reducing display brightness is typically a more favorable user experience than an aggressive display idle timeout.

Windows Vista can control the brightness of the attached display in a mobile PC if the display driver or system BIOS has been properly configured to support operating system control of display brightness. For implementation guidelines, see “Brightness Control in WDDM” on the WHDC Web site. [*[source]*](#_Power_Policy_Optimization)

If the system supports brightness control within Windows, the power policy can specify a default brightness setting. The brightness level is expressed as a percentage of maximum brightness, as shown in Table 3‑3.

Table 3-3 Details for Display Brightness

| **Detail** | **Description** |
| --- | --- |
| Friendly name  | Display brightness |
| Description  | Specifies the brightness level of the display. |
| GUID  | aded5e82-b909-4619-9949-f5d71dac0bcb |
| Defined in | Monitor.inf |
| Balanced defaults  | 100% of maximum brightness (AC)  | 40% of maximum brightness (DC)  |

The observed brightness and power consumption at a given brightness percentage vary from display to display. We encourage system manufacturers to evaluate the brightness of the display on their mobile PC systems and then make appropriate adjustments to the default display brightness in power policy.

## Hard Disk Drive Idle Timeout

Although the hard disk drive (HDD) is not the primary power consumer in the typical mobile PC, power savings can be realized by spinning down the HDD media. The HDD idle timeout allows Windows to automatically spin down the HDD media after a period of disk read and write inactivity.

The power savings that are realized when the HDD media is spun down varies by the make and model of the HDD. We encourage system manufacturers to work with HDD vendors to determine the optimal HDD idle timeout value for specific devices.

By default, Windows Vista specifies moderately long HDD idle timeout values. System manufacturers should consider specifying shorter values when trying to achieve aggressive battery conservation on mobile PCs. Table 3-4 summarizes details of the HDD idle settings.

Table 3-4 Details for HDD Idle

| **Detail** | **Description** |
| --- | --- |
| Friendly name  | Turn off hard disk after |
| Description  | Specifies how long the hard drive is inactive before the disk turns off. |
| PowerCfg Alias  | DISKIDLE |
| Group policy path | Administrative Templates\System\Power Management\Hard Disk Settings\Turn Off the Hard Disk |
| GUID  | 6738e2c4-e8a5-4a42-b16a-e040e769756e |
| Defined in | Ntpoapi.h |
| Balanced defaults  | 60 minutes (AC)  | 30 minutes (DC)  |

## 802.11 Wireless Power-Save Mode

In modern mobile PC platforms, 802.11 wireless devices are almost ubiquitous, and these devices incorporate power-management capabilities. Windows Vista and later versions provide native support for configuring the power-save mode of 802.11 wireless network adapters in power policy.

By default, Windows Vista configures the 802.11 power-save mode to Maximum Performance for both AC and battery (DC) power. This configuration does not allow extra power savings, but does alleviate compatibility problems between some wireless adapters and those access points that are not compatible with 802.11 power-save modes. Table 3‑5 shows the power-save modes and related power policy settings.

Table 3-5 802.11 Power-Save Modes and Power Policy Settings

| **802.11 power-save modes** | **Power policy value** | **Description** |
| --- | --- | --- |
| Maximum Performance | 0 | Specifies maximum performance, in that the 802.11 adapter runs with power-save mode disabled.  |
| Low Power Saving | 1 | Specifies a power save polling (PSP) mode that uses the fastest power-saving mode. |
| Medium Power Saving | 2 | Specifies a PSP mode that favors power savings over performance. |
| Maximum Power Saving | 3 | Specifies a power-save mode with maximum power savings. This mode can be proprietary to the manufacturer of the client 802.11 adapter. |

To realize additional power savings and help extend mobile PC battery life, system manufacturers should consult with the manufacturer of the wireless adapter about changing the power policy value to Maximum Power Saving or Medium Power Saving.

Table 3‑6 lists details for the 802.11 power-save mode power policy in Windows Vista.

Table 3-6 Windows Vista 802.11 Power-Save Mode Power Policy

| **Detail** | **Description** |
| --- | --- |
| Friendly name  | Power-saving mode |
| Description  | Controls the power-saving mode of wireless adapters. |
| GUID  | 12bbebe6-58d6-4636-95bb-3217ef867c1a |
| Defined in | Netwifi.inf |
| Balanced defaults  | Maximum performance (AC) | Maximum performance (DC) |

## Idle Detection Threshold

If a sleep idle timeout is enabled in power policy, Windows Vista automatically places the computer in the sleep state after a period of inactivity. The idle detection threshold deter­mines the degree of processor idle time that is required before the system automatically enters sleep.

The idle detection threshold configures the minimum degree of processor idle time (in percent) that is required for Windows to accrue time toward the sleep idle timeout. By default, the idle detection threshold is configured to 80 percent, indicating that the processor must be 80-percent idle for the power manager to automatically place the system into the sleep state.

Table 3-7 Details for Idle Detection Threshold

| **Detail** | **Description** |
| --- | --- |
| Friendly name  | Idle detection threshold |
| Description  | Required processor idle percentage to sleep. |
| GUID  | 81cd32e0-7833-44f3-8737-7081f38d1f70 |
| Symbol | GUID\_SLEEP\_IDLE\_THRESHOLD |
| Defined in | Ntpoapi.h |
| Balanced defaults  | 80% (AC) | 80% (DC) |

System manufacturers can adjust the idle detection threshold to a lower value to allow the power manager to be more aggressive in automatically transitioning the system to the sleep state. Setting the idle detection threshold to 0 percent is the most aggressive value for power savings and indicates to the power manager that processor activity should be ignored in determining whether the system is idle enough to transition to the sleep state.

## SATA AHCI Link Power Management

Systems running the Microsoft SATA storage driver (Msahci.sys) can benefit from enabling AHCI link power management. The *Intel Serial ATA Advanced Host Controller Interface (AHCI) Specification* defines two AHCI link power-management types that allow the AHCI link to enter low-power modes opportunistically as the link becomes more or less idle: [[*source*](#_Power_Policy_Optimization)]

* Device-initiated power management (DIPM)
* Host-initiated power management (HIPM)

By default, Windows Vista enables HIPM but does not enable DIPM because of device compatibility issues. The registry key described in Table 3‑8 enables DIPM, which can help the AHCI link realize more power savings than HIPM. The power savings gain varies by the make and model of the SATA device that is attached to the link. For more information, see the Microsoft Knowledge Base article 932079. [[*source*](#_Power_Policy_Optimization)]

System manufacturers who use the Microsoft SATA storage driver should add this registry key to their preload images and validate correct disk device operation, concentrating on sleep and resume transition reliability.

Table 3-8 Registry Settings for DIPM for AHCI Links

| **Registry** | **Value** |
| --- | --- |
| Registry Key | HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\ msahci\Controller0 |
| Value Name | DIPMEnabled |
| Value Type | REG\_BINARY |
| Value | 0x01 |

## Interlaced DVD Playback

Beginning with Windows Vista SP1, the Microsoft-provided DVD playback stack renders 60-fields-per-second interlaced DVD content at a rate of 60 frames per second (fps). This high level of DVD playback performance has been shown to significantly strain battery life on some mobile computers when playing this type of content. There is no equivalent issue for progressive content.

System manufacturers can use a registry key added in Windows Vista SP1 to help extend mobile PC battery life by reducing the quality of video playback. This adjustment to video quality has been shown to improve battery life by up to 10 percent, but results will vary significantly across different hardware and device platforms. As with any policy setting, you should validate the policy on the impacted system with preinstalled software loaded.

To reduce the playback rate to 30 fps for interlaced content, you must:

* Disable hardware decode acceleration for DVD playback.
* Enable software deinterlacing, and thereby reduce the playback rate for interlaced video to 30 fps.

To disable hardware decode acceleration for DVD playback, change the registry key and value described in Table 3‑9.

Table 3-9 Registry Setting for Disabling Hardware Decode Acceleration

| **Registry** | **Description** |
| --- | --- |
| Registry Key | HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\scrunch\codecpack\msdvd\ |
| Value Name | DisableDVDDXVA |
| Value Type | REG\_DWORD |
| New Value | 0x01 (default is 0x0) |

To enable software deinterlacing and reduce playback rate for interlaced content to 30 fps, change the registry key and value described in Table 3‑10.

Table 3-10 Registry Setting for Enabling Software Deinterlacing and Reduced Playback Rate

| **Registry** | **Description** |
| --- | --- |
| Registry Key | HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\scrunch\codecpack\msdvd\ |
| Value Name | DeinterlaceMode |
| Value Type | REG\_DWORD |
| New Value | 0x01 (default is 0x0) |

# Optimizing Processor Power-Management Policy

Processors today continue to represent a significant portion of the overall system power budget, while also providing a rich set of power-management controls that offer significant power-saving opportunities. The emergence of multicore and multithreaded processor designs has introduced new complexities into platform and operating system support for processor power management (PPM).

Windows power policy includes values to control how the system uses and applies PPM states. Windows Vista and Windows Server® 2008 include several changes to processor power policy from previous versions of Windows, featuring a simplified and easily understood yet more flexible way to direct the operating system’s use of processor performance states.

Note: A large number of PPM hardware technologies and associated Windows power policy settings exist, including issues for multiprocessor systems. We encourage system manufacturers to review “Processor Power Management in Windows Vista and Windows Server 2008” on the WHDC Web site and consult with their processor vendor before making specific processor power policy changes. As always, any changes to power policy must be validated. [*[source]*](#_Power_Policy_Optimization)

## Favoring Power over Performance

To help extend battery life, processor policy settings can be changed to favor processor power savings over performance. These policy configuration changes optimize the operating system to give a preference to power savings, trading off performance to help extend laptop battery life.

Table 3‑11 describes PPM policy changes that favor power savings over performance. We encourage system manufacturers to make these changes for the default power plan (“Balanced”) when the system is on battery power, as well as for the Power Saver plan, both when the system is on AC (utility) and DC (battery) power.

Table 3‑11 Processor Power Policies for Power Efficiency Optimization

|  |  |  |
| --- | --- | --- |
| Policy | Recommendation | Comment |
| Processor idle state thresholds(C‑state) | * Reduce the C2 time check value to 30 ms
* Lower the C2 pro­mote percentage to 30%
 | Causes C3 idle state to be entered more aggressively when the system is idle. |
| P-state increase (processor perfor-mance state parameter) | Set to SINGLE | Increases performance state one at a time. |
| P-state decrease (processor perfor-mance state parameter) | Set to ROCKET | Decreases performance state to lowest state. |
| P-state increase percentage | Set to 50% | Requires the system to be at least 50% busy before increasing to the next performance state. The Windows kernel limits this value to 80%. The default for the Power Saver scheme is 30%. |
| P-state increase time | Set to 500 ms | Requires the system to be at least 50% busy for 500 ms before increasing to the next performance state. |
| Domain idle accounting  | Set to disabled (off) | Determines how the power manager calculates the target processor performance state in systems with >1 logical processor.* When enabled (the default), this setting favors perfor­mance by choosing the highest performance state of any processor in the domain as the target performance state for all processors in the domain.
* When disabled, this setting favors battery life by calculating the performance state for each CPU independently.
 |

## Reducing Maximum Performance State

Another opportunity to reduce power consumption and help extend battery life is to reduce the maximum processor performance state when the system is on battery power. By default, Windows adjusts processor performance throughout all available processor performance states, according to workload.

For example, suppose a mobile PC processor has the performance states and frequencies shown in Table 3‑12.

Table 3-12 Example System with Maximum Processor State Set to 100%

| State | Frequency (MHz) | Percent | Type |  |
| --- | --- | --- | --- | --- |
| 0 | 2800 | 100 | Performance | **⇦** Maximum Processor State |
| 1 | 2520 |  90 | Performance |  |
| 2 | 2380 |  85 | Performance |  |
| 3 | 2100 |  75 | Performance |  |
| 4 | 1680 |  60 | Performance |  |
| 5 | 1400 |  50 | Performance | **⇦** Minimum Processor State |

The default behavior for the Balanced power plan on battery power is to allow the processor to automatically use any of the possible performance states based on workload. It is possible that if the processor is very busy, the maximum performance state can be used with an associated increase in power consumption.

By reducing the percentage for the maximum processor state, system manufacturers can cap the maximum processor power consumption (and also maximum performance) to help extend battery life. The example in Table 3‑13 demonstrates the same system with the maximum processor state setting configured to 75 percent for battery power.

Table 3-13 Example System with Maximum Processor State Set to 75%

| State | Frequency (megahertz—MHz) | Percent | Type |  |
| --- | --- | --- | --- | --- |
| 0 | 2800 | 100 | Performance |  |
| 1 | 2520 |  90 | Performance |  |
| 2 | 2380 |  85 | Performance |  |
| 3 | 2100 |  75 | Performance | **⇦** Maximum Processor State |
| 4 | 1680 |  60 | Performance |  |
| 5 | 1400 |  50 | Performance | **⇦** Minimum Processor State |

Although this approach limits maximum processor performance on the system, it helps to extend battery life by ensuring that the processor cannot use the maximum performance state, even if the system is very busy.

# Power Policy Recommendations for Extended Battery Life

The power policy settings summarized in Table 3‑14 are recommended for maximum mobile PC battery life. As with any Windows configuration change, system manufacturers must validate these settings on their mobile PC platforms with their software preinstallation images.

##### Recommendation for System Manufacturers and IT Pros

* Define appropriate default power policies for both AC and DC power to maximize battery life.

Table 3-14 Power Policy Recommendations for Maximum Battery Life

|  |  |  |
| --- | --- | --- |
| Policy | Recommendation | Comment |
| PPM | See “[Optimizing Processor Power Management Policy](#_Optimizing_Processor_Power)” earlier in Part 3. | Optimizes for power savings over performance. |
| DVD playback | Disable hardware decode acceleration and enable software deinterlacing. | Allows interlaced content to be played back at 30 fps. |
| 802.11 wireless adapter mode | Set to Maximum Power Savings or Medium Power Savings. | Requires vendor to test with wireless adapter in the system. |
| USB selective suspend | Set to Enabled for all power plans, AC/DC values. | Requires vendor to test platform with integrated USB devices. |
| Display idle timeout | Set to 2 minutes for DC (battery) power. | Has a large impact on system power consumption. |
| Display brightness | Set to reduced level from default for battery power to help extend battery life. | — |
| Sleep idle timeout | Set to 10 minutes for DC power, 15 minutes for AC power. | Puts system to sleep after processor has been at or lower than idle threshold value for the duration of the timeout. |
| Hard disk idle timeout | Set to 5 minutes for DC power, 10 minutes for AC power. | — |
| Idle detection threshold | Set to 0%. | Uses only user input and execution state (ES\_SYSTEM\_REQUIRED) references to determine whether the system should idle to sleep. |
| SATA AHCI link power management | Enable DIPM by using registry key. | — |
| PCIe ASPM | Set to Maximum Power Savings for default power plan, as well as DC values in all power plans. | Requires vendor to validate correct functionality of integrated PCIe devices. See “[PCI Express](#_PCI_Express)” in Part 2. |

# Power Policy Optimization References

White papers on the WHDC Web site:

 Brightness Control in WDDM
<http://www.microsoft.com/whdc/device/display/aero/brightness.mspx>

Optimizing Windows Vista Platforms for Energy Efficiency
[http://www.microsoft.com/whdc/system/pnppwr/powermgmt/optimizePower.mspx](http://www.microsoft.com/whdc/system/pnppwr/powermgmt/optimize_Power.mspx)

Provides guidelines for how to choose values for timeouts that are appropriate for your platform.

Power Policy Configuration and Deployment in Windows Vista
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PMpolicy_Vista.mspx>

Processor Power Management in Windows Vista and Windows Server 2008
 <http://www.microsoft.com/whdc/system/pnppwr/powermgmt/ProcPowerMgmt.mspx>

Provides these specific guidelines:

* How to use ACPI processor objects to implement BIOS support for processor performance states, processor power states, and processor clock throttling.
* How to use the Windows Reliability and Performance Monitor to view statistical data on processor usage and processor speed.
* How to interpret system event log entries related to platform PPM capabilities and ACPI processor objects.
* How to optimize processor power policy by customizing Windows power policies.

Windows Disk Idle Detection
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/Disk_Idle_Detection.mspx>

Tools on TechNet and MSDN:

PowerCfg Command-Line Options (TechNet Web site)
<http://technet2.microsoft.com/WindowsVista/en/library/1d58b934-f56a-4796-b2df-7be2eb9c03bc1033.mspx?mfr=true>

Use the PowerCfg tool (installed with Windows) to change processor power policy—as described on the TechNet Web site.

PwrTest (WDK documentation)
<http://msdn2.microsoft.com/en-us/library/aa906552.aspx>

Use the PwrTest /PPM scenario to view processor power-management states and to view information about processor power state transitions—as described in the WDK documentation. PwrTest is installed with the WDK at %WinDDK%\tools\acpi\pwrtest\ .

Specification:

*Intel SATA Advanced Host Controller Interface (AHCI) Specification* (Intel Web site)
<http://download.intel.com/technology/serialata/pdf/rev1_2.pdf>

TechNet:

Power consumption may be slightly more than expected on a portable Windows Vista-based computer that uses a SATA hard disk that does not support Host-Initiated Link Power Management
<http://support.microsoft.com/kb/932079>

PART 4:
Preinstalled Software and Energy Efficiency

In this part

Battery Life Impact of Preinstalled Software

Why Preinstalled Software Matters for Battery Life

Understanding Resource Usage

Best Practices for Software Applications

Guidelines for Power-Efficient Applications

High-Resolution Periodic Timers and Power Efficiency

Power Event Monitoring Tool: Displaying Power Event Notices

Recommended Testing for Clean vs. Preinstallation Images

Application Power Management References

# Battery Life Impact of Preinstalled Software

Hardware and devices form the foundation for battery life on a mobile PC platform. However, application and service software must also be efficient to allow the system to achieve optimal battery life. Each software component on the system, including the operating system and value-added applications and services, must conform to basic efficiency guidelines. A single misbehaving application or service can eliminate any energy efficiency gains that the latest processor, devices, or platform hardware achieved.

The best measurement of software efficiency is resource usage. This part of the guide focuses on the basic attributes of good software resource usage. Best practices for energy-efficient applications and services are described, including methods for deferring work when the system is running on battery power. At the end of this part, you’ll also find a brief overview of what to measure with tools such as XPerf that can help you to evaluate software resource usage to find opportunities for improving power efficiency.

## Why Preinstalled Software Matters for Battery Life

A single application or service component can prevent a system from realizing optimal battery life. Although Windows provides many power policy configuration options, many systems have poorly optimized preinstalled software or power policy settings that have not been optimized for the host hardware platform.

A common method for evaluating the battery-life impact of preinstalled software is to compare the power consumption of the system with a clean installation of Windows versus a Windows installation that includes value-added software and services. Although a clean installation does not represent the typical platform that OEMs ship to customers, the power consumption comparison can provide insight into the energy-efficiency attributes of preinstalled software.

We conducted an extensive study, with testing performed by a third-party company, to gather data about battery life for several OEM platforms. The tests compared battery-life rundown time of the release-to-manufacturing version of Windows Vista versus OEM preinstallation images, including the latest OEM-available device drivers. As summarized in Figure 4‑1, the data from this study indicates that, in most cases, preinstalled software has a significant negative impact on battery life.

Figure 4‑1 Comparative Difference in Battery Life for Idle Systems

Out of 17 systems tested, 6 systems showed an improvement in battery life with the OEM preinstallation image over the Windows Vista clean installation image, with one system showing significant improvements (>10 percent). Of the 11 systems that regressed in idle battery life, 6 systems showed regressions that were greater than 10 percent.

We partnered with each of the OEMs to complete detailed investigations to find the factors that contribute to the differences in battery life shown in Figure 4‑1. The primary factors are:

* Increased processor, memory, and disk resource usage that could be attributed to value-added applications and services installed on the system.
* Device drivers that were not optimized for the tested hardware platform.
* Unoptimized power policy settings.

The following sections explore how to apply the findings from this study to gain greater power efficiency in application software—especially software that will be preinstalled on OEM systems that run Windows Vista.

## Understanding Resource Usage

Resource usage is used to approximate the impact that an application has on battery life, because it is difficult to associate system power consumption directly with a single application or service. For example, an application that requires a large amount of processor time has a substantial negative impact on battery life. Similarly, an application that periodically writes to the disk can prevent the disk from spinning down to conserve power, and thus may also negatively impacts battery life.

The most common resource usage metrics for applications and services with respect to battery life are processor (CPU), graphics processor (GPU), and disk usage.

### CPU and Power Efficiency

Modern processors are equipped with intricate power-management mechanisms to enable energy efficiency. These technologies are designed to enable performance on demand and extremely low power consumption when the processor is idle. Therefore, the focus for application and service software is to reduce processor usage whenever possible, specifically when the system is idle and the user is not interacting with the computer.

Although some processor power-management technologies are processor vendor-specific, two primary power-management mechanisms are of interest:

* Performance (P-states)

Performance states allow the processor to switch between several discrete voltage and frequency combinations. To save power, a lower frequency is used when the system is idle.

* Idle states (C-states)

Idle states allow large portions of the processor to be powered off in the very small amount of time between clock interrupts—typically 15.6 ms in duration—and other platform hardware events. If the processor is active more often than the clock interrupts, the benefit of the idle state decreases.

To achieve good battery life on a mobile platform, all processor power-management technologies must be used efficiently. Efficient use of the technologies implies that:

* The processor uses the greatest power-saving performance state whenever possible.
* The processor often enters deep idle states.
* Processor usage is kept to the minimum that all applications require.

Software can play a significant role in enabling or preventing the processor from using these power-saving technologies:

* An application might unnecessarily consume processor time by polling for status instead of using event-driven mechanisms.
* A software component might use a high-resolution periodic timer, such as those started with the timeBeginPeriod API, which increases the platform timer tick frequency and prevents effective use of processor idle states.
* A software component might do periodic work when the system is idle, and this work causes the processor to exit the lowest performance state and thus prevents long battery life.
* A software component might perform noncritical housekeeping tasks—such as a scheduled full-disk virus scan—when the system is on battery power and thus prevent long battery life.

Software should be evaluated for processor usage by using inbox utilities such as Reliability and Performance Monitor or advanced tools including XPerf, as discussed in [Part 5](#Part5) of this guide. Process Explorer, which is available from Microsoft TechNet, can show the high-resolution CPU usage of each process on Windows Vista. [*[source]*](#_Application_Power_Management)

##### Recommendation for System Manufacturers—CPU and Power Efficiency

* Target 98-percent average processor idle time over a 10‑minute period of idle.

Although software should be inspected for processor usage across many battery-life test workloads, reduced usage at idle is critical to achieving optimal battery life.

### GPU and Power Efficiency

Graphics device usage also plays a critical role in achieving optimal battery life on a mobile platform. Power-management technologies for graphics devices are similar to those for CPUs, but technologies for graphics devices are typically proprietary and vary based on GPU vendor and product.

##### Recommendation for System Manufacturers—GPU Resource Usage

* Follow the same guidance as for CPUs.

The fundamental guidance for GPU resource usage is the same as for processors. Applications might not intend to impact GPU power or prevent GPU power-management technologies, but might do so inadvertently by:

* Periodically updating the display by causing an otherwise static window to be redrawn unnecessarily.
* Using animations, especially animations that run even when a user is not actively interacting with the application.

### Disk I/O Usage and Power Efficiency

Software components can increase the power consumption of disk devices by unnecessarily writing or reading file or registry data. In addition, frequent intermittent operations can prevent the disk from powering down.

Disk idle detection is key for efforts to achieve maximum mobile PC battery life, because the difference in power consumption between the spin-up and spin-down states can be up to 500  mW on mobile PC form-factor disk devices. This difference is even larger on full-size desktop PC form-factor hard disk devices.

The Windows power manager uses disk activity and power policy to determine when to spin down and when to spin up the disk device. An application can prevent disk spin-down by issuing intermittent file operations (read or write) that are not required to accomplish a user task. Again, the impact is critically important during the idle workload.

Application developers should validate that their applications do not cause disk file operations when the system is idle. Reliability and Performance Monitor or XPerf can be used to inspect for disk activity at a high level or in deep detail, respectively. For information how to determine when the disk is idle, see “Windows Disk Idle Detection” on the WHDC Web site. [*[source]*](#_Application_Power_Management)

##### Recommendation for System Manufacturers—Disk I/O

* Ensure that the software in the preload image allows the disk to spin down for at least 75 percent of an idle period one hour or greater.

# Best Practices for Software Applications

Your design for a Windows-based application should consider that the application will run on many form factors, including mobile PCs, desktop PCs, servers, Tablet PCs, and ultramobile PCs (UMPCs). Many of these form factors are designed to run on battery power, and users expect extended battery life from their systems.

## Guidelines for Power-Efficient Applications

##### Recommendations for Application Developers

To ensure that your application contributes to optimal battery life on Windows platforms, follow these best practices when you design applications or services:

* Invest in performance optimizations.

Every performance optimization is a battery life optimization. Reductions in required resources, such as using less processor time and batching/clustering disk reads, allow system hardware to become idle and enter low-power modes.

* Adjust to user power policy.

Windows Vista makes it easy for the user to choose the overall power-savings or performance behavior of the system. Your application should respond to changes in power policy and reduce resource usage or increase performance accordingly. For example, an application should disable background activity such as indexing or system scanning when the user has selected a Power Saver power plan.

* Reduce resource usage when the system is on battery power.

Your application should reduce its resource usage—such as background update frequency—when the system is on battery power.

* Do not render to the display when it is off.

The system display might be off for power savings. Your application should not perform unnecessary graphics rendering when the display is turned off because this wastes system resources and power. For more information, see “Application Power Management Best Practices for Windows Vista” on the WHDC Web site. [*[source]*](#_Application_Power_Management)

* Avoid polling and spinning in tight loops.

Heavy processor usage reduces the effectiveness of processor power-management technologies such as processor idle states and processor performance states.

* Do not prevent the system from turning off the display or idling to sleep.

Your application should make judicious power requests with the SetThreadExecutionState API. These requests should be made only when critical operations must delay the system from powering off the display or automatically entering sleep.

* Respond to common power-management events.

Your application should register for and respond to common power-management events, such as system power source changes and power-on and power-off notifications for the display. For guidelines, see “Application Power Management Best Practices for Windows Vista” on the WHDC Web site. [*[source]*](#_Application_Power_Management)

* Avoid use of high-resolution periodic timers.

Using high-resolution periodic timers (<10 ms) reduces the efficiency of processor power-management technologies, as discussed in the following section.

* Do not enable debug logging by default. Use Event Tracing for Windows instead.

Periodic debug logging can prevent disk spin-down.

## High-Resolution Periodic Timers and Power Efficiency

The use of high-resolution periodic timers in an application can increase overall system power consumption and subsequently reduce system battery life. You might choose to use high-resolution periodic timers through the Windows Multimedia API—or your application might use these timers inadvertently through external libraries and application development frameworks.

This section provides background information about the use and effects of periodic timers. It includes information about how you can check the use of periodic timers through Windows event logs along with a set of recommendations for best practices to follow in your application design.

### Issues for timeBeginPeriod in Multimedia Applications

The default system-wide timer resolution in Windows is 15.6 ms, which means that every 15.6 ms the operating system receives a clock interrupt from the system timer hardware. When the clock interrupt fires, Windows performs two main actions: it updates the timer tick count if a full tick has elapsed, and it checks whether a scheduled timer object has expired. A timer tick is an abstract notion of elapsed time that Windows uses to consistently track the time of day and thread quantum times. By default, the clock interrupt and timer tick are the same, but Windows or an application can change the clock interrupt period.

Many applications call timeBeginPeriod with a value of 1 to increase the timer resolution to the maximum of 1 ms to support graphical animations, audio playback, or video playback. This not only increases the timer resolution for the application to 1 ms, but also affects the global system timer resolution, because Windows uses at least the highest resolution (that is, the lowest interval) that any application requests. Therefore, if only one application requests a timer resolution of 1 ms, the system timer sets the interval (also called the “system timer tick”) to at least 1 ms. Generally, setting the timer to a value less than 10 ms can negatively impact battery life

Modern processors and chipsets, particularly in mobile platforms, use the idle time between system timer intervals to reduce system power consumption. Various processor and chipset components are placed into low-power idle states between timer intervals. However, these low-power idle states are often ineffective at lowering system power consumption when the system timer interval is less than 10 ms.

If the system timer interval is decreased to less than 10 ms—including when an application calls timeBeginPeriod with a resolution of 1 ms—the low-power idle states are ineffective at reducing system power consumption and system battery life suffers. System battery life might be reduced as much as 25 percent, depending on the hardware platform. The reason is that the transitions to and from low-power states incur an energy cost. Therefore, entering and exiting low power states without spending a minimum amount of time in the low-power states may be more costly than if the system simply remained in the high-power state.

If your application changes the timer resolution, you should ensure that any external libraries or application development frameworks do not unexpectedly also change the system timer interval on the application’s behalf.

You can check the event log to determine whether an application has changed the system timer interval. Windows Vista automatically generates a system event when the system timer interval changes and then logs the process name and the requested interval in the kernel power diagnostic log. The event is event ID 63:

The application or service [path-to-process-image] is attempting to update the system timer resolution to a value of [timer-interval-in-100-ns-units].

For example, a test application named test generates the following event when it calls the timeBeginPeriod API with a value of 1 ms:

The application or service \Device\harddiskVolume1\test.exe is attempting to update the system timer resolution to a value of 1000.

To enable the kernel power diagnostic event log

1. Open the Windows Event Viewer in the Control Panel Administrative Tools application.

2. On the View menu, click Show Analytic and Debug Logs.

3. In the left tree view, navigate to Applications and Services Logs > Microsoft > Windows > Kernel-Power.

4. Right-click Diagnostic and click Enable Log.

5. Restart the system.

After the system is restarted, you can view various events in the kernel power diagnostic event log. Look for event ID 63, which indicates system timer interval change events.

### Best Practices for High-Resolution Periodic Timers

##### Recommendations for Application Developers

In your work designing and developing applications, follow these best practices when using high-resolution periodic timers:

* Understand the impact of high-resolution periodic timers on system power consumption.

A single application can cause the system timer interval to change, which can reduce system battery life as much as 25 percent.

* Validate that your application does not unexpectedly change the system timer interval.

To do this, search for event ID 63 in the Windows Kernel Power diagnostic event log.

* If your application must use a high-resolution periodic timer, enable the periodic timer only while the required functionality is active.

For example, if the high-resolution periodic timer is required for animation, disable the periodic timer when the animation is complete.

* If your application must use a high-resolution periodic timer, consider disabling use of the periodic timer and associated functionality when a Power Saver power plan is active or when the system is on battery power.

# Power Event Monitoring Tool: Displaying Power Event Notices

The Power Event Monitoring tool is provided as working example code for developers who want to integrate support in their products for responding to common power events, including sleep and resume events. This tool:

* Registers for common power events—including system power source and power plan personality change notifications—and displays a message when the notifications occur.
* Displays current battery capacity information.
* Allows you to interact with SetThreadExecutionState and execution state flags by manipulating settings in the application.
* Provides code for application development that demonstrates the use of various power-management APIs.

You can run this tool to gain a better understanding of when certain power events are delivered. The tool calls the RegisterPowerSettingNotification API to register for the following power events, which are displayed when notifications occur:

* Away Mode transitions
* System power source changes
* Power plan personality changes
* Background task notifications
* Battery capacity remaining notifications
* Monitor power state changes

The tool also receives and displays messages for the following system sleep and resume events:

PBT\_APMSUSPEND
PBT\_APMRESUMEAUTOMATIC
PBT\_APMRESUMESUSPEND

To build and use the Power Event Monitoring tool

1. Download the package from the WHDC Web site on a Windows Vista system where the Windows Vista SDK and Visual Studio 2005 are installed.

For more information on the Power Event Monitoring tool, see “Power Event Monitoring Tool Sample Application” on the WHDC Web site. [*[source]*](#_Application_Power_Management)

You must have the latest Windows SDK libraries installed on your system. You might also need to add the Windows SDK include path to the Visual Studio include directories list.

To install the SDK, see “Installation Instructions for the Windows SDK” on MSDN. [*[source]*](#_Application_Power_Management)

2. Open the Visual Studio 2005 Project file in the package and, from the **Build** menu, click Build Solution to compile it into the Microsoft Power Event Monitoring utility.

Note: Although building this tool with different compilers and build environments might be possible, Microsoft does not support other build procedures.

3. Launch the newly built application from the Visual Studio environment.

At the bottom of the main window are check boxes that correspond to the different execution state flags that can be set when using SetThreadExecutionState. If you select any of these check boxes, this tool calls SetThreadExecutionState with flags that correspond to the checked items.

For more information:

* For detailed documentation, see “Application Power Management Best Practices for Windows Vista” on the WHDC Web site. [*[source]*](#_Application_Power_Management)
* To provide feedback to Microsoft or for technical assistance, contact: onnow@microsoft.com

# Recommended Testing for Clean vs. Preinstallation Images

We recommend that you create baseline tests and run them on both a clean Windows installation and your planned preinstallation image to understand the impact of preinstalled software.

To do this, use tools such as XPerf to evaluate the usage of the various components in the system, as described in [Part 5](#Part5) of this guide. A suggested test methodology for measuring battery life is outlined in [Appendix A](#AppA) of this guide.

##### Recommendation for System Manufacturers and IT Pros

* Use the testing and analysis tools to understand which activities in the preinstallation image might cause the CPU, GPU, and hard drive to consume unnecessary power.

Top issues to check in testing are summarized in Table 4-1.

Table 4-1 Issues for Testing Preinstalled Software

| Issue | Testing notes |
| --- | --- |
| Processor idle time check | * Confirm that the processor has 98% idle time over a 30-minute period with the preinstallation image—starting 10 minutes after boot to allow for post-boot activity to complete.
* Ensure that default software that can be configured or changed—such as default sidebar gadgets—does not reduce the system’s ability to achieve the target goal of 98% idle time.
 |
| Frequent activity check | * Ensure that frequent periodic thread or DPC activity does not occur with the preinstallation image. The smallest period should be greater than 100 ms.
* Ensure that applications in the preinstallation image do not make frequent Windows Management Instrumentation (WMI) calls, which can prevent the system from entering and exiting lower power states.
* Ensure that applications in the preinstallation image use polling only if absolutely necessary—the application solution should instead be interrupt or event driven.
* Ensure that applications in the preinstallation image do not make frequent disk or registry accesses.
 |
| DC versus AC operation check | * Check that applications in the preinstallation image maximize every opportunity to scale back activity when the system runs on DC versus AC power.
* Ensure that the preinstallation correctly integrates software applications into the operating system power policy, as described in “Application Power Management Best Practices for Windows Vista” on the WHDC Web site.[*[source]*](#_Application_Power_Management)
 |
| Sleep and idle check | * Validate that applications do not prevent the display and the system from going idle and entering sleep.
* Ensure that the default timer tick frequency is set to 15.6 ms when the system is idle.
 |
| Task tray applications check | * For preinstalled applications that are always running (such as those that display task-tray icons), ensure that those applications use unmanaged, lightweight code. These applications should:— Avoid using Windows Presentation Foundation (WPF) for task-tray applications.— Avoid animations in general, especially when there is no user input on the system, and especially avoid animated task tray icons.
* Ensure that update utilities for preinstalled application do not attempt to install updates while the system is running on battery power.
* Inspect any scheduled tasks that applications install. Ensure that such tasks are scheduled to run only when the system is on AC power.
 |
| Boot and shutdown times check | * Evaluate the impact of preinstalled application on boot and shutdown times by comparing with a clean Windows installation on the same hardware.
* Ensure that only required services are run at system startup and that optional services are not launched by default.
 |

# Application Power-Management References

White Papers and Topics on MSDN:

Application Power Management Best Practices for Windows Vista
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PM_apps.mspx>

Summary of Usability Guidelines (MSDN)
<http://msdn2.microsoft.com/en-us/library/ms695506.aspx>

Tools:

Power Event Monitoring Tool Sample Application (WHDC Web site)
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PM-apps_samp.mspx>

Documentation in “Application Power Management Best Practices for Windows Vista” on the WHDC Web site

Process Explorer

<http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx>

Windows Software Development Kit (SDK)
<http://msdn2.microsoft.com/en-us/library/ms717328.aspx>

PART 5:
Evaluating and Measuring Battery Life

In this part

Getting Started with Measuring Energy Efficiency

Choosing Scenarios to Measure

Idle Test Scenario

DVD Playback Test Scenario

Applications in Test Scenarios

Benchmarks and Battery Life

General Performance and Analysis Tools from Microsoft

Event Tracing for Windows

Windows Reliability and Performance Monitor: Viewing Processor Usage

Xperf: Measuring and Analyzing System Performance

PwrTest: Viewing Supported PPM States and Transitions

Testing Tools Reference

# Getting Started with Measuring Energy Efficiency

In [Part 5](#Part5) of this guide, we describe how to define test scenarios and then examine different techniques to measure energy efficiency. We examine the important issues to consider when measuring battery life and how to evaluate the results to determine appropriate solutions for managing energy efficiency.

We recommend that system manufacturers choose a combination of measurement methods, because it is not possible to adequately evaluate the factors that influence battery life with a single measure. Table 5‑1 summarizes the basic methods of measuring specific elements of energy efficiency and power consumption, along with advantages and disadvantages of each method.

For a step-by-step description of how to create a basic battery life test methodology for multiple system configurations and multiple Windows versions, see [Appendix A](#AppA) of this guide.

Table 5‑1 Methods for Measuring Energy Efficiency

| Measure  | Advantages | Disadvantages |
| --- | --- | --- |
| **System power consumption**  |
| 1. Sample instantaneous power to measure AC and DC con­sump­tion.2. Then, average power consumption over time and over repeated scenario.See Figure 5-1. | * Measures actual system power consumption.
* Can be used to approx­i­mate battery life.
* Divide battery capacity by average power consumption to get actual battery life.
* Easy to mea­sure across differ­ent hardware configurations.
 |  |
| **Direct battery** |
| Use battery run-down tests:* Subject system to a specific workload on battery power.
* Measure time required to drain battery from full to empty.
* Repeat and average the duration times.
 | * Provides authentic reflection of actual user experience.
* Easy to generate data across various hard­ware configurations.
 | * Requires excessive time to run multiple test iterations across systems.
* Highly sensitive to BIOS changes, device drivers, battery parameters, and so forth.
* Requires large invest­ment in hardware.
 |
| **Instrumented systems** |
| Use specialized measure­ment hardware and software for analysis.See Figure 5-2. | * Renders extremely granular data.
* Individual components can be measured, such as HDD, CPU, memory, GPU, audio, and so forth.
* Impact of small code changes can be evaluated.
 | * Requires custom hardware—expensive investment.
* Does not scale broadly.
 |
| **System and device resource usage** |
| **1.** To evaluate power consump­tion, examine:* Processor and disk activity.
* P‑state and C‑state instrumentation.
* Other I/O activity, such as interrupts or DMA.

**2.** Evaluate device-specific resource usage:* USB scheduler activity.
* Frequent DMA impacts processor idle state.
* Network usage.
* Platform timer tick frequency.

See “[Xperf: Measuring and Analyzing System Performance](#_Xperf:_Measuring_and)” later in Part 5. | * Does not require special power-measurement tools.
* Resource usage analysis tools such as Xperf are available.
 | * Hard to validate battery life impact of small code changes.
* No insight into hard­ware-specific power consumption problems — for example, 802.11 power mode stuck on High Performance.
 |
| **Distribution of activity over time** |
| Examine CPU and disk activity over time.See “[PwrTest: Viewing Supported PPM States and Transitions](#_PwrTest:_Viewing_Supported)” later in Part 5. | * PwrTest /PPM can be used to view and log P‑state and C‑state information over a specified interval.
* PwrTest /DISK can be used to ensure the disk can turn off after a period of inactivity.
 | * Hard to validate battery life impact of small code changes.
* Hard to identify root cause of problems without use of tools such as Xperf.
 |
| **Frequency changes with Event Tracing for Windows events** |
| Measure processor frequency changes dynamically:* P‑states.
* C‑state change notifications via PowerProcessorIdle­State events.

See “[Windows Reliability and Performance Monitor: Viewing Processor Usage](#_Windows_Reliability_and)” later in Part 5. | * ETW is enabled by default on all systems.
* Reliability and Performance Monitor can be used to view and log changes.
 | * Hard to identify specific applications that prevent lower power states.
 |



Figure 5‑1 Test Configuration for Direct Power Measurement



Figure 5‑2 Test Configuration for Instrumented Systems

# Choosing Scenarios to Measure

Evaluating power consumption through a scenario-based approach can help ensure that key targeted scenarios are optimized on your platform. You should carefully design scenarios and workloads that reflect what end users typically do, because it is critical to ensure that you optimize the scenarios that typical users care about.

At a minimum, idle and DVD playback are two scenarios that should be evaluated. Idle power consumption should be optimized because power consumption during the idle scenario will be seen across all other workloads. DVD playback is another key scenario that many end users care about, and it exercises many different components in the operating system and at the hardware level.

#### Characteristics to Consider in Test Scenarios

Consider these characteristics when defining the scenarios in which you will measure the energy efficiency of mobile PC systems:

* Does the scenario accurately reflect the activities that a user typically does while the system runs on battery?
* Does the scenario include a range of common situations, such as:
* Running commonly used applications.
* Typing at human speeds.
* Using multiple applications.
* Switching and moving windows.
* Using representative data-sets.
* Using documents of appropriate sizes, running a typical number of media items, and so on.

#### What to Avoid in Test Scenarios

Conversely, the scenarios you select should avoid these issues:

* Avoid measuring the impact of hardware alone.

Scenarios should consider the impact that both the software and hardware have on battery life.

* Avoid skewing toward operations that are solely CPU intensive or disk intensive.

A well-thought-out scenario exercises different components in the system to understand how each component impacts battery life.

#### Workloads to Consider in Test Scenarios

Performance in each scenario will vary across the range of hardware and the installed software, and for different form factors and operating environments. When you select workloads to include in your measures, consider these factors:

* Does this workload operate the same way on each system?
Or does it change based on the hardware and software in the configuration?
* Is this workload disabled in some configurations?
* How does behavior change in the workload when the system is on AC versus DC power?

We use several different workloads to measure and evaluate systems for energy efficiency, focusing on workloads relevant to mobile PC users when the system is running on battery, as shown in Table 5‑2 and summarized in the following sections.

Table 5-2 Common Workload Scenarios for Energy Optimization

| Workload | Scenario description |
| --- | --- |
| Idle | Represents the baseline for system activity and all other workload scenarios.  |
| DVD playback | Represents common user scenarios such as watching a movie on a plane. This scenario is also used in common battery-life benchmarks. |
| Office productivity | Represents the typical business user with elements such as typing, document creation, and e‑mail as well as a media-rich scenario. |

## Idle Test Scenario

Mobile systems are typically idle for significant time periods—while users are sitting in meetings, reading documents, and so on. Therefore, ensuring that your system is optimized for the idle scenario can have a significant impact on overall battery life. Correctly setting idle timeouts in Windows Vista can significantly improve battery life—by as much as 10 percent in our testing.

Because the idle scenario represents the baseline for all system activity, the difference in battery life when testing the idle scenario should generally be reflected across every user scenario.

To analyze whether components interfere with a system’s ability to switch to idle states

1. Create a standard methodology to compare preinstallation images versus Windows Vista “clean installation,” as described in [Part 5](#Part5) and [Appendix A](#AppA) of this guide.

2. To evaluate which components keep the system in active states and thus have the greatest negative impact on energy efficiency, run PwrTest to create logs for these specific scenarios: /BATTERY, /DISK, and /PPM.

For details, see “[PwrTest: Viewing Supported PPM States and Transitions](#_PwrTest:_Viewing_Supported)” later in Part 5.

3. To analyze specific system and device resource usage, run Xperf.

For details, see “[Xperf: Measuring and Analyzing System Performance](#_Xperf:_Measuring_and)” later in Part 5.

## DVD Playback Test Scenario

DVD playback is a popular usage scenario on mobile systems and is included in many benchmarks. It exercises many different aspects of the system, including optical drive, CPU, and GPU and therefore is a key scenario that should be optimized.

In Windows Vista SP1, a registry key is available to allow system manufacturers to change the frame rate that is used when the system plays back interlaced DVD content with the Windows Media® Player codec. The registry setting allows customers who use the Windows Media Player codec to render interlaced content at 30 fps, compared to the default setting of 60 fps. The adjustment to video quality has been shown to improve battery life by up to 10 percent, but results will vary for specific system configurations. For details, see [Part 3](#Part3) of this document.

## Applications in Test Scenarios

This section describes the basic steps for analyzing how applications affect battery life or sleep and resume performance. These scenarios are intended to simulate what typical users may do on mobile systems to ensure that the most widely used operations are optimized for your platform.

At Microsoft, we evaluate two categories of applications in defining these scenarios:

* Productivity
Microsoft Office Word, Microsoft Excel®, Microsoft Outlook®, and Acrobat Reader. Such applications allow you to evaluate the effects of common activities such as typing at human speeds, switching windows, and so on.
* Media Experience
Windows Media Player, Windows Mail, and similar interactive and media-rich applications.

To evaluate application impact on mobile system power efficiency

1. Define a standard methodology to compare your preinstallation image versus the Windows Vista “clean installation,” including your selection of productivity and media-rich and communications applications.

2. Run your standard stress tests with these applications on your mobile system in conjunction with your standard battery run-down test.

3. Use PwrTest to automate sleep and resume transitions, and observe whether the user-element applications interfere with aggressive sleep states.

4. With the preinstallation image that includes your value-added applications, run the same standard stress tests and repeat the use of PwrTest to automate sleep and resume.

Again, observe whether any preinstalled applications interfere with sleep and resume performance.

5. Use Xperf to better understand application-specific activity impact on resource usage.

For details, see “[Xperf: Measuring and Analyzing System Performance](#_Xperf:_Measuring_and)” later in Part 5.

# Benchmarks and Battery Life

Benchmarking to evaluate battery life or other performance measures can be difficult because the operating system attempts to improve its performance over time based on observed usage patterns.

Windows Vista presents even greater challenges than earlier versions of Windows because the system’s behavior changes when user input is detected, when battery power states change, and based on other default and user-defined policies.

Therefore, any benchmark methodology must account for Windows behavior that represents “real-world” behaviors. We strongly recommend that you train the system before running your benchmark. Training ensures that results reflect real-world usage and increases the ability to reproduce consistent results.

We recommend the following methodology when creating benchmarks, with the goal of measuring something similar to actual system performance in real-world scenarios:

1. Install
Install the operating system and the benchmarking software.

2. Train the system
Run the benchmark at least once to train SuperFetch™.
—or —
Use the applications in the benchmark as real users would.

3. Let idle tasks run
Leave the system alone after using it to allow all idle tasks to run (up to 3 days for disk layout optimizations).
—or —
Call the ProcessIdleTask API from Advapi32.dll to force idle tasks to run.

4. Test
Run the benchmark.

Table 5-3 provides brief notes on issues to consider in defining any benchmark methodology that you use with your battery performance measures. For details, see “Measuring Performance in Windows Vista” on the WHDC Web site. [*[source]*](#_Testing_Tools_Reference)

Table 5-3 Tips for Benchmarking

| Area | Tip |
| --- | --- |
| System services | Do not disable system services for most measurements.  |
| User Account Control (UAC) elevation prompts | Disable UAC prompts to simplify the benchmark script. Caution: Do not disable UAC in a production or personal work environment. |
| Windows Error Reporting (WER) prompts | Set WER to queue all errors, which minimizes impact on benchmarking results. |
| Scheduled tasks | In addition to the preceding idle tasks, many systems include tasks that are scheduled to run at specific times of the day. To ensure consistent benchmark results, be aware of any scheduled tasks that might run during the benchmark. Either disable them or ensure that they run during every benchmark. |
| Memory management and SuperFetch | * Windows Vista memory management is based on observation, prefetching, and prioritization, which can affect efforts to benchmark workloads. To ensure that your workload measurements are consistent, you must account for how SuperFetch adapts to user behavior. This means training SuperFetch before measuring and tracking how closely each training run matches the measured workload.
* Before training the system, the initial run of the test might exhibit lower performance due to increased I/O. You should generally not average measurements from such a run with measurements of subsequent runs.
 |
| Volume Shadow Copy service | Volume Shadow Copy is not predictable and can impact benchmarks, so you should consider disabling the service. To compare Windows XP and Windows Vista, we recommend either enabling Volume Shadow Copy for both versions or disabling it for both. |
| Delaying activity after system boot | When you run workloads, delay the workload start until post-boot activities are finished. Wait 7 minutes after boot before running benchmark workloads to reduce the variability caused by post-boot activities. |
| Post-processing results | Copy traces or logs from the test machine to another location for any post-processing. Delete them from the machine before running benchmark tests again. |
| Battery conditioning | When running battery run-down tests, condition the battery before starting your actual testing by charging and discharging the battery until consistent battery life numbers are achieved. |

# General Performance and Analysis Tools from Microsoft

Several common performance analysis tools can be used specifically to observe and measure power-management behavior in Windows systems. All these tools take advantage of Event Tracing for Windows (ETW). In general, the tools provide different levels of detail depending on how detailed the analysis needs to be:

* Windows Reliability and Performance Monitor
This tool provides information about the use of various components as well as processor power states.
* Xperf
This tool provides additional information over the Windows Reliability and Performance Monitor to better analyze specific power-related issues. In addition, XPerf can help analyze specific problems that are related to application start times, boot times, and interrupt activity and can provide more detailed information on resource usage.

The following sections provide information about tools that can be used to help analyze issues related to battery life.

## Event Tracing for Windows

Event Tracing for Windows (ETW) is a tracing system that has been widely used since it was first introduced in Windows 2000. Because of its flexibility, ETW is the instrumentation method of choice for the Windows platform and is a key tool for performance analysis.

We assume that you are basically familiar with ETW. Some key ETW features that are beneficial for energy efficiency measurement include the following:

* The Windows kernel and many other Windows components are already instrumented, so system behavior can be viewed by processing and examining kernel events. In addition, the behavior of any instrumented component or application can be correlated with system behavior.
* All events are disabled by default, and the call to EventWrite() incurs only a flag check. Thus, the code you ship with your systems can be fully instrumented without incurring performance overhead or exposing proprietary intellectual property, and tracing can be enabled at any time in customers’ systems.
* An ETW logging session has a defined maximum memory footprint. The disk footprint can be similarly limited. CPU load is directly proportional to the event rate and scales very well. Hence, event tracing has a very low runtime overhead.

## Windows Reliability and Performance Monitor: Viewing Processor Usage

The Windows Reliability and Performance Monitor (which includes PerfMon) provides a summary view of processor usage versus processor speed, as shown in Figure 5-3. You can use Reliability and Performance Monitor to examine how programs affect a computer’s performance, both in real time and by collecting log data for later analysis.



System in low P‑state

Low CPU usage

Figure 5-3 Reliability and Performance Monitor Showing Performance State Usage

Windows Vista provides performance counters that you can use to view processor speed changes in response to power policy changes or to view increases in processor usage when running a policy that uses demand-based switching (DBS) for performance states. Application developers can enable the use of performance counters within their application to help evaluate application-specific performance. By using these performance counters in Reliability and Performance Monitor, you can see the current processor speed for each processor or core present in the system, as shown in Figure 5-4.



Figure 5-4 Performance State Usage Viewed in Reliability and Performance Monitor

In Windows Server 2008 and Windows Vista SP1, the Windows power manager publishes kernel processor power events to the System event log and includes error events if ACPI processor objects fail validation during driver initialization.

Reliability and Performance Monitor can also be used to view and log the system’s use of processor power states or APCI C‑states. You can use the Histogram view to compare system idleness to the three possible C‑states on a system, as shown in Figure 5‑5.

To run Reliability and Performance Monitor to look at C states

1. On the **Start** menu, in the **Start Search** box, type:
perfmon

2. In the far-left pane, click Performance Monitor.

3. Right-click the graph and click Add Counters.

4. In the Available Counters, expand the **Processor** group counters.

5. Click %C1, %C2, and % C3 Time.

6. In the **Instances of selected object** box, click \_Total to ensure that you capture the total time across all processors.

7. Right-click one of the counters at the bottom and, on the **Graph** tab, click **Histogram** for the view.



Figure 5-5 Reliability and Performance Monitor Showing C‑State Usage

To run Reliability and Performance Monitor

1. On the **Start** menu, in the **Start Search** box, type:
perfmon

2. Follow the steps in Reliability and Performance Monitor Help to add processor performance counters, as shown in Figure 5-6.



Figure 5-6 Adding Processor Performance Counters in Reliablity and Performance Monitor

For more information:

* For a detailed walkthrough that uses Reliability and Performance Monitor to measure processor performance, see “Processor Power Management in Windows Vista and Windows Server 2008” on the WHDC Web site. [*[source]*](#_Testing_Tools_Reference)

Extensive documentation is available in the local Windows Vista help under the topic “Performance Monitoring for IT Pros.”

* Information is provided in “Windows Vista Performance and Reliability Monitoring Step-by-Step Guide” in the TechNet library. [*[source]*](#_Testing_Tools_Reference)

## Xperf: Measuring and Analyzing System Performance

The Performance Explorer (Xperf) tools are new to the Windows Server 2008 Software Development Kit (SDK) and are designed to measure and analyze Windows system and application performance. These tools are useful to a broad audience including system builders, hardware manufacturers, driver developers, and software developers. You can use Xperf to analyze a wide range of performance problems, including application start times, boot issues, deferred procedure call and interrupt activity, system responsiveness issues, application resource usage, and interrupt storms.

The tools use a trace capture and analysis model that follows this general flow:

1. Enable tracing by using Xperf.exe.

2. Run the workload.

3. Disable tracing and save the data to a trace file by using Xperf.

4. Then, further process or view the trace files.

To use Xperf for analyzing power (example)

1. Start NT Kernel Logger with latency and power flags:

xperf -on latency+power

2. Start the user- mode trace power logger:

xperf -start power -on Microsoft-Windows-Kernel-Power -f po.etl

3. Run the workload.

4. Stop the kernel trace, and then stop the user-mode trace:

xperf –stop

xperf -stop power

5. Merge the two traces into one trace called powerTrace.etl:

xperf -merge \kernel.etl po.etl powerTrace.etl

6. Analyze the trace:

xperf powerTrace.etl

7. Examine resource usage for CPU and disk across all workloads through the graphical interface of Xperf to gain a better understanding of the processes that are running.

8. Examine P-state information from the trace:

xperf -I powertrace.etl -o powertrace.txt

grep PowerProcessorPerfState powertrace.txt

For more information:

* Performance Explorer tools are available through the Windows Server 2008 SDK. [*[source]*](#_Testing_Tools_Reference)
* For more information, including links to online forums, FAQs, and more, see the MSDN home page. [*[source]*](#_Testing_Tools_Reference)

## PwrTest: Viewing Supported PPM States and Transitions

The power-management test tool (PwrTest) can exercise and record system power-management information. You can use PwrTest to automate sleep and resume transitions and record power management and battery information over a period of time. PwrTest can run in either a sampling mode that summarizes usage over time or an event-driven mode that logs transitions that the Windows power manager initiates.

PwrTest provides several command-line options to tailor the sampling rate and information logged. You can run PwrTest for specific scenarios by using the following switches:

|  |  |
| --- | --- |
| /BATTERY  | Displays and records battery information and metrics. Battery capacity, voltage, rate of drain, and estimated life time remaining can be logged for a period of time at a specified interval. Scenario details are available in “PwrTest Battery Scenario” in the WDK. [*[source]*](#_Testing_Tools_Reference) |
| /DISK | Displays the current status of disk idle detection on the system, so that system manufacturers can validate that the disk can be powered off on systems with the preinstallation image installed. Scenario details are available in “Windows Disk Idle Detection” on the WHDC Web site. [*[source]*](#_Testing_Tools_Reference) |
| /ES  | Displays and records thread execution state changes. Thread execution states enable applications and services to temporarily override power-management settings such as the monitor and sleep idle timeouts. Scenario details are available in “PwrTest Execution State Scenario” in the WDK. [*[source]*](#_Testing_Tools_Reference) |
| /INFO  | Displays system power-management information, such as available sleep states and processor power-management capabilities. Scenario details are available in “PwrTest Info Scenario” in the WDK. [*[source]*](#_Testing_Tools_Reference) |
| /PPM  | Displays and records processor power-management information and metrics. Performance (P states) and processor idle (C states) state information can be logged for a period of time at a specified interval. Scenario details are available in “PwrTest PPM Scenario” in the WDK. [*[source]*](#_Testing_Tools_Reference) |
| /SLEEP  | Exercises sleep and resume transition functionality. Sleep and resume transitions can be automated, and target sleep states can be specified. Scenario details are available in PwrTest Sleep Scenario in the WDK. [*[source]*](#_Testing_Tools_Reference) |

# Testing Tools Reference

White Papers:

Measuring Performance in Windows Vista <http://www.microsoft.com/whdc/system/sysperf/Vista_perf.mspx>

Processor Power Management in Windows Vista and Windows Server 2008
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/ProcPowerMgmt.mspx>

Tools and Kits:

Event Tracing (MSDN Library)
<http://msdn2.microsoft.com/en-us/library/aa363787.aspx>

EventWrite() Function (MSDN Library)
[http://msdn2.microsoft.com/en-us/library/aa363752.aspx](http://msdn2.microsoft.com/en-us/library/aa363752.aspx%20)

User Interface: Windows Reliability and Performance Monitor (TechNet Library)
<http://technet2.microsoft.com/WindowsVista/f/?en/library/8669ff6c-6c52-43fe-86ed-a4fccb6091e81033.mspx>

Performance Explorer tools and information (MSDN home page)
<http://go.microsoft.com/fwlink/?LinkId=103276>

Power Event Monitoring Tool Sample Application (WHDC Web site)
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PM-apps_samp.mspx>

Command-line reference A-Z (for Windows Server, TechNet Library)
<http://technet2.microsoft.com/WindowsServer/en/library/552ed70a-208d-48c4-8da8-2e27b530eac71033.mspx?mfr=true>

User Interface: Windows Reliability and Performance Monitor (TechNet Library)
<http://technet2.microsoft.com/WindowsVista/f/?en/library/8669ff6c-6c52-43fe-86ed-a4fccb6091e81033.mspx>

Using Reliability Monitor
PerfMon – Performance Monitoring: Performance counting, performance logs and alerts, and System Monitor documentation (TechNet Library)
<http://technet2.microsoft.com/WindowsVista/en/library/d33c3cc1-a9dd-4b96-8062-edb21ee70f0e1033.mspx?mfr=true>

Windows Software Development Kit for Windows Vista and .NET Framework 3.0 Runtime Components
<http://www.microsoft.com/downloads/details.aspx?FamilyID=c2b1e300-f358-4523-b479-f53d234cdccf&DisplayLang=en>
Check this site for availability of the Windows Server 2008 SDK.

Performance Counters for ASP.NET (MSDN library)
<http://msdn2.microsoft.com/en-us/library/fxk122b4.aspx>

WPP Software Tracing (Windows software trace preprocessor, WDK documentation )
 [http://msdn2.microsoft.com/en-us/library/ms793164.aspx](%20http%3A/msdn2.microsoft.com/en-us/library/ms793164.aspx)

Windows Vista Performance and Reliability Monitoring Step-by-Step Guide (TechNet Library)
<http://technet2.microsoft.com/WindowsVista/en/library/ab3b2cfc-b177-43ec-8a4d-0bfac62d88961033.mspx>

Microsoft Developer Network (home page)
<http://go.microsoft.com/fwlink/?LinkId=103276>

Scenarios:

PwrTest (WDK documentation)
<http://msdn2.microsoft.com/en-us/library/aa906552.aspx>

PwrTest Battery Scenario (WDK documentation)
<http://msdn2.microsoft.com/en-us/library/aa906541.aspx>

Pwrtest /DISK scenario information in “Windows Disk Idle Detection” (WHDC Web site)
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/Disk_Idle_Detection.mspx>

PwrTest Execution State Scenario (WDK documentation)
<http://msdn2.microsoft.com/en-us/library/aa906553.aspx>

PwrTest Info Scenario (WDK documentation)
<http://msdn2.microsoft.com/en-us/library/aa906543.aspx>

PwrTest PPM Scenario (WDK documentation)
<http://msdn2.microsoft.com/en-us/library/aa906546.aspx>

PwrTest Sleep Scenario (WDK documentation)
<http://msdn2.microsoft.com/en-us/library/aa906550.aspx>

APPENDIX A:
Test Methodology for Measuring Battery Life

In this appendix

Principles for Testing Battery Life

Overview of Testing Methodology

Creating a Baseline Configuration for Testing

Configuring Hardware and Operating System for Testing

Conditioning and Training the System before Testing

Running the Battery Life Tests

Evaluating Changes

# Principles for Testing Battery Life

This appendix describes a master methodology for testing battery life for multiple system configurations and multiple Windows versions. By establishing a standard baseline test methodology, you can compare battery life in different configurations and over time.

This methodology uses two different tests for battery life:

* Idle: A test in which the mobile platform is powered on and then sits idle.
* DVD playback: A test in which the mobile platform plays a DVD movie.

Consider these fundamental principles for testing:

* Test each system configuration in the same states, with at least these minimum Windows images:
* An OEM image created by the mobile platform vendor.
* A clean base Windows Vista Ultimate image.

Note: The term “clean base image” refers to the retail version of Windows. The retail Windows Vista image is commonly referred to as the shrink-wrapped boxed product or fully packaged product (FPP).

* Use the same settings for each installation of Windows Vista whenever possible—that is, only change settings directly related to the scenario you are testing.
* Set up the mobile platform with the default settings whenever possible to reflect the experience that users will have with the default configurations.

If you plan to change the default power settings when you ship the platform, then ensure that both the clean base configuration and the OEM image use the same settings.

* Update each system's BIOS, if possible, before testing.

# Overview of Testing Methodology

For each Windows version or configuration on which you plan to run the battery tests, perform the following steps.

1. Create a baseline system configuration for testing.

2. Configure the hardware and power settings for testing.

3. Create an image of the configured system so that you can distribute it to the other mobile platforms that you plan to test.

4. Record the following system configuration details:

* Display brightness.
* Power policy settings.
* Any registry changes that were made related to power policy (for example, the Interlaced DVD playback registry setting described in [Part 3](#Part3) of this guide).
* Driver versions.
* Which codecs are being used during DVD playback.

5. Condition both the battery and the system.

6. Run the tests.

7. Repeat Steps 2–7 when evaluating the preinstallation image that you want to compare.

The following sections describe each of these steps in detail.

# Creating a Baseline Configuration for Testing

The baseline system configuration consists of the Windows operating system version that you plan to test, any current updates to that software, and the required drivers and their updates.

To create a baseline configuration for testing

1. Install a clean version of Windows.

2. Install current Windows updates.

3. Update drivers if required.

## Installing a Clean Version of Windows

The first step in creating a baseline configuration is to install a clean version of Windows. This section describes how to install a clean version of Windows Vista Ultimate. Install your Windows Vista OEM image by using your standard methods.

To create a baseline Windows Vista Ultimate configuration, you must have a Windows Vista Ultimate DVD.

To install Windows Vista Ultimate

1. Place a Windows Vista Ultimate DVD in the drive, and turn on the system power.

2. When prompted, press any key to boot from CD or DVD.

3. At the language and preference screen, accept the default options, and click Next.

4. Click Install now.

5. Do not enter a Product key. Clear the Automatically Activate Windows when I’m online check box, and click Next.

6. At the Do you want to enter your product key now screen, click No.

7. Select **Windows Vista Ultimate**, select the I have selected the edition of Windows that I purchased check box, and then click Next.

8. Check I accept the license terms, and click Next.

9. At the Which type of installation do you want screen, select Custom (advanced).

10. Click Drive options (advanced).

11. Select the disk on which you want to install Windows Vista Ultimate, and click Format.

12. At the All data stored will be permanently deleted dialog box, click OK.

13. After the system finishes formatting the disk, click Next.

14. Leave the password blank. Type User in the Username field, and click Next.

15. Type a computer name and click Next.

16. Click Use recommended settings.

17. Select the correct time zone, date, and time, and click Next.

18. At the Select your computer’s current location screen, click Public location.

19. At the Thank you screen, click Start.

20. When the Windows Vista installation completes, close the **Welcome Center** dialog box.

21. At the Set network location screen, click Public location, and then click Close.

22. Install any OEM-specific drivers according to the manufacturer’s instructions.

## Installing Current Windows Updates

Regardless of the Windows version that you are planning to test, you should install all current updates as part of the baseline configuration.

To install Windows Updates

1. When the PC system is connected to the Internet, click Start > All Programs > Windows Update.

2. Click View available updates. If no updates are available for the current Windows version, you will see a message that says Windows is up to date.

3. Select all available updates, and click Install.

4. At the Windows needs your permission dialog box, click Continue.

5. When the updates complete, close the Windows Update window.

## Installing Updated Drivers

After you install updates from the Windows Update Web site, you should make sure that you have the latest drivers for all the devices on the system.

To install drivers

1. Click Start, and right-click Computer.

2. Select Manage from the drop-down menu.

3. Click Continue if the system presents a **User Account Control** dialog box.

4. Select Device Manager.

5. A yellow exclamation point appears next to each device with a driver issue.

6. For each device-driver issue, go to the Web site of the system vendor and search for Windows driver updates for the Windows version that you are configuring.

7. If no applicable device drivers are available at that Web site, go to the Web site of the device vendor and install any driver updates available for that component.

8. Record all driver updates that you install so that you can replicate the configuration later if required for additional testing.

9. After drivers have been updated, run the Windows Experience Index to ensure that the system reflects any updated driver capabilities.

# Configuring Hardware and Operating System for Testing

This test methodology measures a baseline battery life with the operating system and minimal additional device usage. In calculating battery life, it does not consider network traffic, screen saver use, or power-management sleep states. Factors such as network traffic might not be deterministic and therefore might prevent you from getting repeatable results.

Before you start testing:

* Disable the following:
* Network connections and external devices
* The screen saver
* The display idle timeout
* The sleep idle timeout
* Automatic updates and scheduled events, such as system restore
* UAC elevation prompts
* System Restore
* Volume Shadow Copy Service
* Force idle background tasks to execute.
* Be aware of any scheduled tasks that might run during the benchmark testing.
* Leave Aero Glass at the setting that Windows Vista decides is appropriate after completing the Windows Experience Assessment.
* Ensure that Search indexing is enabled, which is the default.
* Apply the same default power policy configuration in the scenarios that you plan to use during battery life evaluation.

In some specific scenarios, it might be necessary to change additional individual power settings, such as Domain Idle Accounting, Wireless Power Save, and Scaling the output frame rate when playing Interlaced DVD content. These settings are documented in “Optimizing Windows Vista Platforms for Energy Efficiency” on the WHDC Web site at:
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/optimize_Power.mspx>

After you have a baseline battery life estimate, you can set up additional tests that are tailored to your usage scenarios. Such tests might include representative network traffic, screen saver usage, and occasional sleep states so that you can estimate battery life under such a load.

## Disabling Network Connections and External Devices

Disable all types of network connections and any external devices, such as USB and IEEE 1394 devices, according to the instructions that follow.

To disable network connections and external devices

1. Disable each of the following hardware components if it is not already off:

* Wireless (use the physical switch if one is available)
* Bluetooth wireless technology
* IR

2. Disconnect any external devices.

3. Do not plug in the network adapter.

4. Leave all services in their default states.

## Disabling the Screensaver

To disable the screensaver and power management on Windows Vista

1. Right-click the desktop and select Personalize.

2. Click Screensaver.

3. Select None from the drop-down menu, and click Apply.

4. Click Change power settings.

5. Under Balanced, click Change plan settings.

6. Select Never from both drop-down menus next to the following items:

* Turn off the display.
* Put the computer to sleep.

7. Click Save changes.

8. Close any open windows.

9. Ensure that each system is set to the default power scheme, which is the Balanced power scheme for Windows Vista.

10. Do not minimize the taskbar.

## Disabling the Display Idle Timeout

The display idle timeout configures the period of user inactivity before Windows turns off the display to save power. We recommend that you set the value to 0 for battery life evaluation, which disables the display idle timeout.

To facilitate automation, use the PowerCfg command-line utility to disable the display idle timeout.

To disable display idle timeout

* Use the following PowerCfg command:

C:\>powercfg -setdcvalueindex SCHEME\_CURRENT SUB\_VIDEO VIDEOIDLE 0

C:\>powercfg -setactive SCHEME\_CURRENT

## Disabling the Sleep Idle Timeout

The sleep idle timeout configures the period of user inactivity before Windows automatically places the system into sleep (the ACPI S3 state). We recommend that you set the value to 0, which disables the sleep idle timeout for battery life evaluation.

To facilitate automation, use the PowerCfg command line utility to disable the sleep idle timeout.

To disable sleep idle timeout

* Use the following PowerCfg command:

C:\>powercfg -setdcvalueindex SCHEME\_CURRENT SUB\_SLEEP STANDBYIDLE 0

C:\>powercfg -setactive SCHEME\_CURRENT

## Disabling the Hibernate Idle Timeout

The hibernate idle timeout configures the period of time after sleep before Windows wakes up and automatically places the system into hibernate (the ACPI S4 state). We recommend that you set the value to 0, which disables the hibernate idle timeout for battery life evaluation.

To facilitate automation, use the PowerCfg command-line utility to disable the hibernate idle timeout.

To disable sleep idle timeout

* Use the following PowerCfg command:

C:\>powercfg -setdcvalueindex SCHEME\_CURRENT SUB\_SLEEP HIBERNATEIDLE 0

C:\>powercfg -setactive SCHEME\_CURRENT

## Disabling Automatic Windows Updates

We recommend that you disable Windows Updates during battery life evaluation.

To disable automatic updates in Windows Vista

1. Click Start, right-click Computer, and select Properties.

2. In the **See Also** group, click Windows Update.

3. Click Change settings.

4 Select Never check for updates, and then click OK.

5. At the User Account Control dialog box, click Continue.

## User Account Control Elevation Prompts

UAC allows Windows Vista to make a much firmer distinction between actions undertaken by an ordinary user and actions that require administrator privileges. Generally, Windows Vista runs at the least privileged access level and prompts the user for confirmation if a requested action requires more elevated privileges.

For added security, the elevation prompt is designed to be difficult to script. For benchmarking, you can take two approaches to avoid UAC prompts:

* Run the workloads as the built-in administrator, which requires that the account be explicitly enabled after Setup.
* Allow elevation of privileges without prompting by disabling UAC prompts.

Recommended: Disable UAC prompts to simplify the benchmark script.

Caution: Do not disable UAC in a production or personal work environment.

To disable UAC prompts

* In an elevated Command Prompt window, run the following command (all on one line):

reg.exe ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion \Policies\System /v EnableLUA /t REG\_DWORD /d 0 /f

See also the blog entry “Windows Vista Secret #4: Disabling UAC” at <http://blogs.msdn.com/tims/archive/2006/09/20/763275.aspx>

## Disabling System Restore (Optional)

We recommend that you disable system restore during battery life evaluation.

To disable system restore on Windows Vista

1. Click Start, right-click Computer, and then select Properties.

2. Under Tasks in the left panel, click System protection.

3. At the User Account Control dialog box, click Continue.

4. Under Automatic restore points, clear all the check boxes.

5. Click Apply and OK, and then close both windows.

## Disabling Volume Shadow Copy Service (Optional)

Volume Shadow Copy Service simplifies and integrates the backup and restoration of both system and user files. The operating system takes snapshots of disk volumes at regular intervals, before application installations, and before system updates. When a snapshot is created, most sectors in a volume are marked as copy-on-write and can be updated only after a copy has been made. The copies are of two forms:

* Sectors that are frequently updated are copied by a low-priority background process (background “precopy”).
* All other sectors are copied on write as a synchronous process.

Both the background precopy and the synchronous copy-on-write processes can affect performance. The background writes cause random I/O that, although done at low priority, can impose a seek penalty on foreground sequential I/O streams and, for disks that have less effective dynamic random access memory (DRAM) cache management, can force foreground reads to wait for lower priority background writes. Windows Vista throttles the precopy process when the user is present, pausing 100 ms between copies.

Benchmarking Tip: Because Volume Shadow Copy is not predictable and can impact benchmarks, you might want to consider disabling the service.

## Executing Idle Tasks

Windows Vista file placement optimization, which is done at most once every 3 days, is an example of a task that is carried out when the system is idle. System Restore and other features of Windows Vista also attempt to defer some work until the system is idle. Some work items that are done once after Windows Setup runs also operate under the idle task scheduling mechanisms.

You can force execution of these idle tasks by executing the ProcessIdleTasks API. The purpose of the API is to give benchmarks a simple way to force any pending idle tasks to be executed immediately, without being required to wait a long time. This helps ensure repeatable results. In some cases, if the objective is to evaluate the impact of these idle tasks, then it might be acceptable to not force these tasks to complete immediately.

However, for the purpose of ongoing test measurements, we recommend that you force these tasks to complete immediately.

You can call the API ProcessIdleTasks in one of two ways: from the command line or synchronously from a program.

To call ProcessIdleTasks from the command line

* Use the following syntax:

Rundll32.exe advapi32.dll,ProcessIdleTasks

When called from the command line, the ProcessIdleTasks work occurs asynchronously in the background and can take 10 to 15 minutes to complete. Task Manager reports which processes are running, and the disk is likely active during this time.

To call ProcessIdleTasks from a program

* Use the ProcessIdleTasks API, which takes no arguments and returns a standard Microsoft Win32® error code, as follows:

DWORD

ProcessIdleTasks()

The API itself is synchronous, so it does not return until the idle tasks have completed.

## Scheduled Tasks

In addition to the preceding idle tasks, many systems include tasks that are scheduled to run at specific times of the day, whereas others are performed when the system is idle. To ensure consistency of your benchmarks, be aware of any scheduled tasks that might run during the benchmark. We do not recommend that you disable these tasks, but it is important to understand when these tasks are scheduled to ensure consistency across test runs. Some of these scheduled tasks include:

* Windows Defender (the Windows Vista antispyware technology) periodically scans the system and monitors the registry and areas of Windows in real time for changes that might be caused by malicious software.

Windows Defender uses low-priority I/O.

* Disk defragmentation moves fragmented files to larger contiguous regions on the disk. It moves files that are accessed at boot and application startup toward the outer edge of the disk, where higher transfer rates can be achieved.

Disk defragmentation also uses low-priority I/O.

Benchmarking Tip: Be aware of your scheduled tasks and how they might impact benchmarks run at different times of the day. If you find that they impact your results, delete the tasks or reschedule them to avoid conflict with your benchmark runs.

# Conditioning and Training the System before Testing

Laptop batteries require some level of conditioning on first use to ensure maximum life span of the battery.

To condition the battery

1. Plug the system into an external power source.

2. Fully recharge the battery.

3. Turn the system on and wait for the system to boot.

4. Wait until the system’s battery runs down and the system goes into standby.

5. Record the length of time required to fully discharge the battery.

6. Repeat the process of charging and discharging until you get consistent times across 2 to 3 runs.

Windows Vista takes a proactive approach to memory management by using the SuperFetch service to observe system usage over time and to prefetch useful pages into memory. Windows XP used prefetching to speed boot and application start, but Windows Vista uses this mechanism much more broadly to bring programs, documents, and other data into memory before their actual use. Prefetched data remains on the system’s standby memory list, which has been reorganized and redesigned to make it easier to preserve useful data in memory over longer periods of time. Windows Vista introduces a standby list with priorities that allows it to keep historically important pages in memory versus the recently used ones (such as a recently copied file that might not be used again).

As a machine boots, large amounts of code and data are read from the disk. Systems with more than 1 gigabyte (GB) of RAM set aside large amounts of memory for a cache of prefetched data. After boot the system releases this cache, its pages become available, and SuperFetch fills these pages with new contents. Because some of the post-boot activities throttle themselves or use low-priority mechanisms, their impact on interactive users is controlled. However, they can be a source of variation in benchmarking measurements.

Benchmarking Tip: When you run workloads, delay the workload start until these post-boot activities are finished. Wait 7 minutes after boot before running benchmark workloads to reduce the variability caused by post-boot activities

To train the system

1. Run the benchmark at least once before taking measurements to allow SuperFetch to prefetch the relevant data.

2. Wait 7 minutes before starting the battery life tests.

# Running the Battery Life Tests

Follow these guidelines for running the battery life tests:

* Run the idle battery-life test and the DVD playback test at least three times on each test system under each operating system configuration.
* Provide an external way to verify the rundown time of each battery. A simple approach is to use a video camera that captures the system and a clock.
* For the DVD playback test, make sure that Windows Media Player is set to repeat if playback completes before the battery is depleted.

## Running the Idle Battery-Life Test

Repeat the following steps three times for each system.

To run the idle battery-life test—repeat 3 times

1. Plug the system into the external power source and fully recharge the battery.

2. Turn the system on and wait for the system to fully boot.

3. Verify that no window is in focus.

4. Ensure that a method is in place to track the results of the test—either through a video camera or through software logging.

5. Disconnect the power.

6. Record the start time for this iteration of the idle test through software logging or via the clock that the video camera was recording.

7. Wait until the system’s battery runs down and the system shuts down. Record that time as the end time for the test.

## Running the DVD Playback Test

Repeat the following steps three times for each system and for each Windows configuration.

To run the DVD playback test—repeat 3 times

1. Plug the system into the external power source and fully recharge the battery.

2. Turn the system on and wait for the system to fully boot.

3. Insert a DVD into the DVD drive and close the DVD drive.

4. The DVD will start in Windows Media Player. Make sure that Windows Media Player is set to repeat if playback completes before the battery is depleted

5. Put the DVD player into full-screen mode.

6. Ensure that a method is in place to track the results of the test—either through a video camera or software logging.

7. Disconnect the power from the laptop.

8. Record the start time for this iteration of the DVD playback test through software logging or via the clock that the video camera was recording.

9. Wait until the system’s battery runs down and the system shuts down. Record that time as the end time for the test.

# Evaluating Changes

After you have captured a baseline configuration and results for both the Idle and DVD playback tests, you can better understand and evaluate the effect of any system or configuration changes on battery life.

You should retest the systems after changing any of the following parameters, and compare those results against the baseline configuration:

* Updated drivers
* Changes in codecs
* Changes in power policy settings
* Changes in operating system images including additional preinstalled software.