Processor Power Management in Windows 7 and Windows Server 2008 R2

January 15, 2010

Abstract

Windows® 7 and Windows Server® 2008 R2 include updated support for ACPI processor power management (PPM) features, including support for processor performance states and processor idle sleep states on multiprocessor systems. This paper provides details of the support in Windows 7 and Windows Server 2008 R2, describes how PPM functions with the Windows 7 and Windows Server 2008 R2 power policy store, and provides guidelines for firmware developers and system designers.

The information in this white paper applies to the following operating systems:
 Windows Server 2008 R2
 Windows 7

The current version of this paper is maintained on the Web at:
<http://www.microsoft.com/whdc/system/pnppwr/powermgmt/ProcPowerMgmtWin7.mspx>

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

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

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

© 2010 Microsoft Corporation. All rights reserved.

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| January 15, 2010 | First Publication |

Contents

[Introduction 4](#_Toc251244760)

[Processor Power Management Changes in Windows 7 4](#_Toc251244761)

[Windows 7 Processor Power Management Functional Overview 5](#_Toc251244762)

[Fundamental Components 5](#_Toc251244763)

[System Firmware (BIOS) 6](#_Toc251244764)

[Windows Kernel Power Manager 7](#_Toc251244765)

[Processor Drivers 7](#_Toc251244766)

[Windows 7 Processor Driver Support 8](#_Toc251244767)

[The Generic Processor Driver 8](#_Toc251244768)

[Implementing Processor Power Management for Windows 7 8](#_Toc251244769)

[Operating System Validation of ACPI Processor Objects 10](#_Toc251244770)

[Declaring Processors 10](#_Toc251244771)

[Implementing BIOS Support for Processor Performance States 10](#_Toc251244772)

[Required Performance State ACPI Objects 10](#_Toc251244773)

[Optional Performance State ACPI Objects 10](#_Toc251244774)

[Describing Processor Control Dependencies 11](#_Toc251244775)

[Implementing BIOS Support for Processor Idle States 12](#_Toc251244776)

[C‑state Enumeration 12](#_Toc251244777)

[Implementing BIOS Support for Processor Clock Throttling 13](#_Toc251244778)

[T‑state Enumeration 13](#_Toc251244779)

[Core Parking 14](#_Toc251244780)

[Processor Clocking Control 15](#_Toc251244781)

[Processor Power Management Validation Tools 16](#_Toc251244782)

[Windows Resource Monitor 17](#_Toc251244783)

[Windows Reliability and Performance Monitor 17](#_Toc251244784)

[Viewing Processor Performance State Usage 18](#_Toc251244785)

[Viewing Processor Idle State Usage 20](#_Toc251244786)

[Event Viewer 21](#_Toc251244787)

[PowerCfg 23](#_Toc251244788)

[Pwrtest 23](#_Toc251244789)

[Viewing Supported PPM States 23](#_Toc251244790)

[Viewing and Logging Processor Idle State Transitions 23](#_Toc251244791)

[Processor Power Policy on Windows 25](#_Toc251244792)

[Usage Guidelines 25](#_Toc251244793)

[Firmware Implementation of Processor Power Policy 26](#_Toc251244794)

[Windows 7 Power Policy Store 26](#_Toc251244795)

[Processor Power Policy Settings 27](#_Toc251244796)

[Processor Performance Policy Settings 27](#_Toc251244797)

[Processor Core Parking Policy Settings 31](#_Toc251244798)

[Processor Idle Policy Settings 37](#_Toc251244799)

[System Cooling Policy Setting 39](#_Toc251244800)

[Next Steps 39](#_Toc251244801)

[Resources 40](#_Toc251244802)

# Introduction

Saving power in today’s computers is a primary design goal, whether to prolong battery life and manage system thermals in portable computers or to address growing energy consumption in desktop and server systems. The processors in today's computers 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.

Windows® 7 and Windows Server® 2008 R2 include updated and enhanced support for processor power management (PPM) technologies, including Core Parking, improvements to PPM policy configuration, and cooperative processor performance control with the underlying hardware. This paper provides details about the PPM implementation in Windows 7 and Windows Server 2008 R2 that are useful for system designers, firmware engineers, and test engineers.

This paper is an updated and expanded version of ”Processor Power Management in Windows Vista and Windows Server 2008*.*” We encourage you to review the Windows Vista® version of this paper for details about how to implement PPM support for Windows Vista and Windows Server 2008.

**Note:** The remainder of this paper refers to both Windows 7 and Windows Server 2008 R2 collectively as Windows 7.

## Processor Power Management Changes in Windows 7

Windows 7 includes the following PPM changes and enhancements:

* **Discrete PPM policies**

Each configuration option for PPM has its own independent power policy setting. All PPM power policy settings use the same framework and the same PowerCfg utility interface options for configuration. The grouping of PPM power policy settings into collections of settings and the **/ppmperf**, **/ppmidle**, **/encode,** and **/decode** PowerCfg options are not supported in Windows 7.

* **Core Parking**

The Windows kernel power manager cooperates with the kernel scheduler to intelligently target work to a subset of all logical processors in the system to help improve energy efficiency. On systems that have processors that include Intel Hyper-Threading Technology, Core Parking also helps improve system performance and responsiveness.

* **Improved diagnostic utilities**

The PowerCfg utility has been enhanced with the **/energy** option to diagnose common power management problems, including errors in the firmware that describe a computer's PPM capabilities.

* **Processor Clocking Control (PCC)**

PCC enables cooperative processor performance and throttle state management with the underlying platform firmware. PCC helps the platform enable the underlying power management features, such as fan speed control, while the processor remains within a range of performance states. PPC is supported only on Windows Server 2008 R2.

* **Intelligent Timer Tick Distribution (ITTD)**

ITTD helps to extend the amount of time that processor cores remain in the idle state by not interrupting all cores in the system when the periodic timer interrupt is delivered. Only the base service processor (BSP) receives every timer tick interrupt, which it optionally delivers to secondary processor cores. When ITTD is combined with Core Parking, it helps systems that have Hyper-V installed realize reduced interrupt traffic and longer idle periods.

* **Updated PPM policies**

The default PPM policies in Windows 7 have been updated to improve energy efficiency and to balance system performance with power savings. When a computer is running on battery power, the High Performance power plan uses multiple processor performance states. Similarly, the Power Saver plan uses multiple processor performance states and is no longer capped at the lowest supported performance state. These improvements help balance the power savings with overall system performance.

# Windows 7 Processor Power Management Functional Overview

This section provides an overview of the implementation of the PPM functionality and policy control for Windows 7.

## Fundamental Components

The responsibility for PPM is divided between the platform firmware and components of the operating system as follows:

* System firmware (ACPI BIOS)
* Windows kernel power manager
* Windows processor drivers

Figure shows each of these components. Each component is described in more detail in the following sections.



Figure 1. Windows 7 PPM components

### System Firmware (BIOS)

The system firmware is responsible for including the ACPI BIOS tables that contain the ACPI processor objects that describe the system's PPM capabilities and controls. The ACPI BIOS tables can also contain optional ACPI processor objects that describe multiprocessor domain dependencies or support for cooperative performance state management (PCC). The system firmware can dynamically update the PPM capabilities based on certain system events, such as AC/DC power source transitions.

Details about the required and optional ACPI processor objects for Windows 7 are provided in the “Implementing Processor Power Management for Windows 7” section later in this paper.

### Windows Kernel Power Manager

The Windows kernel power manager is responsible for:

* Managing and applying processor power policy.
* Calculating the required processor power state transitions.
* Applying any constraints due to thermal conditions.

Processor power policy is owned and managed by the Windows kernel power manager. The power manager is responsible for choosing the correct processor state (the “target” state) based on CPU usage and other factors, depending on the PPM technology that is involved. The power manager then directs the appropriate processor state transition through a direct-call interface to the processor driver. The processor driver is responsible for invoking the actual state transition mechanism on the processor or other platform hardware as required.

### Processor Drivers

Processor drivers in Windows are used to abstract the specific differences of processor designs from various processor vendors from the operating system. For example, processor drivers might contain processor-specific routines to determine the presence of PPM capabilities and might use a model-specific register (MSR) to invoke PPM state transitions directly on a processor.

Processor drivers in Windows are responsible for:

* Enumerating the PPM capabilities of the system.
* Validating that the PPM capabilities are described correctly and that they are present in system hardware.
* Passing the supported PPM capabilities to the kernel power manager.
* Invoking processor power state transitions on the appropriate hardware as directed by the kernel power manager.

#### Loading the Correct Processor Driver

Processor objects are described in the ACPI namespace and are enumerated by the Windows ACPI interpreter. Windows uses the information that is obtained from the CPUID instruction to generate the correct hardware device ID during processor enumeration. This enables the system to load the correct processor driver for the processors that are installed in the system. If a specific match is not found, the generic processor driver, *Processr.sys*, is used. The specific processors that Windows supports can be seen by examining the Cpu.inf file that is located in the %SYSTEMDRIVE%\Windows\Inf directory of a Windows installation.

## Windows 7 Processor Driver Support

Table 1 lists the processor drivers that are included in Windows 7.

Table 1. Windows 7 Processor Driver Support

| **CPU vendor** | **Supported processor model/family** | **Driver name** |
| --- | --- | --- |
| AMD | OpteronTurion 64Athlon 64SempronMobile Sempron | Amdk8.sys |
| AMD | Phenom | Amdppm.sys |
| Intel | Pentium 4Mobile Pentium 4 processor that  supports Hyper-Threading TechnologyPentium MCore familyCeleron familyPentium familyAtom familyXeon family | Intelppm.sys |
| VIA | C7-M processor | Viac7.sys |
| - | Any processor not listed above | Processr.sys |

### The Generic Processor Driver

Microsoft provides a generic processor driver, *Processr.sys*, that can be used on any Windows 7–compatible system or processor type, with the following requirements:

* The ACPI 2.0 \_PSS and \_PCT processor objects must be present in the ACPI namespace.
* The \_PCT Performance Control object must describe the PERF\_CTRL and PERF\_STATUS control and status registers in system I/O space.

Optionally, the Microsoft-defined Extended PSS (XPSS) object can be used with the generic processor driver to enable the use of ACPI Functional Fixed Hardware (FFH) controls for the PERF\_CTRL and PERF\_STATUS registers. For more information about implementing XPSS, refer to the “Microsoft Extended PSS ACPI Method Specification” on the WHDC Web site.

**Note:** The processor drivers that provided native PPM support for AMD K7 family processors and Transmeta Crusoe family processors in earlier versions of Windows have been removed from Windows 7.

# Implementing Processor Power Management for Windows 7

Windows 7 includes expanded support for the processor objects that are defined in the ACPI 2.0 and ACPI 3.0 specifications. Several of the processor objects are required for Windows 7 to support processor performance states, whereas other processor objects are optional and can be leveraged at the discretion of the system firmware engineer.

Table lists the ACPI processor objects and indicates which of these objects are supported by the various versions of Windows.

Table 2. Windows ACPI Processor Object Support

|  |  |  |  |
| --- | --- | --- | --- |
| Defined by | ACPI object | \_\_\_\_\_Windows Version\_\_\_\_Windows Windows Windows XP1 Vista2 73 | Comments |
| ACPI 1.0 | FADT P\_LVLx\_LAT, P\_LVL2, P\_LVL3, P\_CNT, P\_BLK,DUTY\_OFFSET, DUTY\_WIDTH | • | • | • |  |
| P\_LVL2\_UP | X | X | X |  |
| FADT PSTATE\_CNT, CSTATE\_CNT | • | • | • |  |
| ACPI 2.0 | \_PSS | • | • | • | Required for processor performance state control |
| \_PCT | • | • | • | Required for processor performance state control |
| \_PPC, Notify(*CPU,* 0X80) | • | • | • | Required for processor performance state control |
| \_CST, Notify(*CPU,* 0X81) | • | • | • | Optional |
| \_PTC | X | • | • | Optional |
| \_PDC | • | • | • | Optional; \_PDC use is vendor-specific (superseded by \_OSC) |
| ACPI 3.0 | \_OSC | X | • | • | Optional; \_OSC use is vendor-specific |
| \_TSS | X | • | • | Optional |
| \_TPC, Notify(*CPU,* 0X82) | X | • | • | Optional |
| \_PSD | X | • | • | Optional |
| \_TSD | X | • | • | Optional |
| \_CSD | X | • | • | Optional |
| Microsoft | XPSS | X | • | • | Optional support that is used only by the generic processor driver |
| PCC | X | X | • | Windows Server 2008 R2 only |

**Legend:**• = Supported
X = Not Supported

1 The ACPI support indicated in this column applies to both Windows XP and Windows Server 2003.

2 The ACPI support indicated in this column applies to both Windows Vista and Windows Server 2008.

3 The ACPI support indicated in this column applies to both Windows 7 and Windows Server 2008 R2.

## Operating System Validation of ACPI Processor Objects

To increase platform reliability and to help OEMs and firmware developers discover and resolve problems, Windows performs validation checks on the ACPI processor objects that are defined in the system firmware during processor driver initialization. If errors, incorrect or ambiguous information, or other problems are detected, Windows disables the relevant PPM control. In this situation, the kernel power manager logs an error in the system event log that describes which processor control (performance states, idle states, or throttle states) were disabled.

Additionally, the results of this validation are made available through the PowerCfg utility when the **/ENERGY** parameter is specified. For more information about how to use the PowerCfg utility for energy efficiency analysis, see “Using PowerCfg to Evaluate System Energy Efficiency” on the WHDC Web site.

## Declaring Processors

For each processor in the system, the ACPI namespace should declare a named processor object by using the Processor operator. Processor objects may appear anywhere under the \\_SB scope or may appear under the \\_PR scope to maintain compatibility with ACPI 1.0 operating systems. Processor objects must be in the \\_SB or \\_PR scope, but not in both.

The processor’s object list may optionally include any of the ACPI processor objects in Table , which appears earlier in this paper. The processor objects that are required to support PPM in Windows are described in the remainder of this paper.

## Implementing BIOS Support for Processor Performance States

This section discusses the ACPI processor objects that are required to support ACPI processor performance states (P‑states). Systems that do not support processor performance states should not include these ACPI objects in the ACPI namespace.

### Required Performance State ACPI Objects

Processor P‑states must be declared in the BIOS by using the ACPI 2.0 performance state objects in the processor’s object list.

The following ACPI objects are required to support processor performance states:

* \_PSS
* \_PCT
* \_PPC

### Optional Performance State ACPI Objects

The following ACPI objects are optional for Windows 7 and Windows Server 2008 R2 to support processor performance states:

* Notify(*CPU*, 0x08)
* \_OSC, \_PDC
* \_PSD
* XPSS

**Note:** For more information about the Microsoft-defined XPSS method and the generic processor driver interface, see “The Generic Processor Driver” earlier in this paper and ”Microsoft Extended PSS ACPI Method Specification” on the WHDC Web site.

### Describing Processor Control Dependencies

System firmware may optionally describe platform control dependencies for processor performance, throttle, or idle states by using the dependency objects that were introduced for this purpose in ACPI 3.0. If present, each dependency object must describe no more than one coordination domain. If the platform supports processor performance states and describes a domain dependency by using \_PSD, the kernel power manager uses the P‑state domain dependency relationship information to control processor linear throttle states.

#### Artificial Processor Performance State Domains

When Windows Vista and Windows Server 2008 were developed, several dual-core and dual-logical processor designs were prevalent in the marketplace. These CPUs typically provide one set of performance state controls that are shared across both cores or both logical processors, which implies that a control dependency exists. However, the system firmware for systems that have these CPUs targets earlier operating systems or otherwise predates the release of Windows Vista and therefore generally does not provide the ACPI 3.0 dependency objects. To support these popular processors without first requiring a platform BIOS update, Windows 7 creates an “artificial” processor performance state dependency domain for the operating system to use. Windows synthesizes a dependency domain for all logical processors in the same physical package. This is the default behavior for Windows 7, and therefore the \_PSD object is not required to be present in the ACPI namespace to realize multiprocessor performance states on systems that are equipped with these dual-core processors.

For future systems that might provide two processor cores that share the same physical package but that have independent PPM controls, the default behavior of creating an artificial control dependency domain can be overridden in the following two ways:

* Include a \_PSD object in the ACPI namespace that describes a separate dependency domain number for each logical processor or for each processor core.
* Create a DWORD value that is named **PerfEnablePackageIdle** under the following registry key and set its value to 1:

 **HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Throttle**

## Implementing BIOS Support for Processor Idle States

This section discusses the ACPI processor objects that are required to support ACPI processor idle states (C‑states). Systems that do not support processor idle states should not include these ACPI objects in the ACPI namespace.

Processor C‑states can be declared in the BIOS by using either of the following ACPI processor idle state control interfaces:

* The ACPI 2.0 \_CST object in the processor’s object list.
* The legacy ACPI 1.0 interface that uses the Processor Register Block (P\_BLK) P\_LVL2 and P\_LVL3 registers and FADT P\_LVLx\_LAT values.

### C‑state Enumeration

The processor driver first looks for support for ACPI 2.0 C‑states in the system firmware, as indicated by the presence of the \_CST and \_CSD objects in the ACPI namespace. If the system does not support ACPI 2.0 C‑states, the driver looks for support for ACPI 1.0 C‑states.

#### ACPI 2.0 C‑state Initialization

During initialization, the processor driver looks for and evaluates the \_CST and \_CSD ACPI objects. If the processor driver finds these objects, it performs validation checks to ensure that these objects are correctly populated and compatible with Windows C‑state support. The following validation checks are performed:

* \_CSD object validation
* The \_CSD object must report that the system supports hardware coordination of C‑states (**CoordType** = 0xFE, HW\_ALL). If any other value is reported for **CoordType**, all C‑states other than C1 are disabled on the system.
* \_CST object validation
* The C-states that are listed in the \_CST object must be ordered with increasing values for:

 C‑state type (C1, C2, and so on)
 Latency
 Power

* All C‑state control register addresses must be located in I/O space.
* All entries must have valid addresses for the required ACPI BIOS interfaces:
	+ The C2 state must have a valid address for the PM1a control block.
	+ The C3 state must have valid addresses for the:

 PM2 control block
 PM1a control block
 PM1a event block

#### ACPI 1.0 C‑state Initialization

If support for ACPI 2.0 C‑states is not present in the system firmware, the processor driver attempts to initialize ACPI 1.0 C‑states. The following validation checks are performed:

* C1 validation
* C1 is enabled on all systems by default.
* C2 validation
* The P\_LVL2\_LAT value that is reported in the FADT must be below 100 µs.
* There must be a valid address for the PM1a control block.
* There must be a valid P\_BLK address.
* The system must be a uniprocessor system. C2 is not supported on multiprocessor systems by default, unless the processor driver explicitly enables this based on known capabilities of the system.
* C3 validation
* The P\_LVL3\_LAT value that is reported in the FADT must be below 1000 µs.
* There must be a valid address for the PM2 control block.
* There must be a valid address for the PM1a control block.
* There must be a valid address for the PM1a event block.
* There must be a valid P\_BLK address.
* The system must be a uniprocessor system. C3 is not supported on multiprocessor systems by default, unless the processor driver explicitly enables this based on known capabilities of the system.

## Implementing BIOS Support for Processor Clock Throttling

This section discusses the ACPI processor objects that are required to support ACPI processor linear stop clock throttle states (T‑states). Systems that do not support processor T‑states should not include these ACPI objects in the ACPI namespace.

Processor T‑states can be declared in the BIOS by using either of the following ACPI processor throttling control interfaces:

* The ACPI 2.0 and 3.0 \_PTC, \_TSS, and \_TPC objects in the processor’s object list.
* The legacy ACPI 1.0 interface that uses the Processor Register Block (P\_BLK) P\_CNT register.

### T‑state Enumeration

The processor driver first looks for support for ACPI 3.0 T‑states in the system firmware, as indicated by the presence of the \_PTC, \_TSS, and \_TPC objects in the ACPI namespace. If the system does not support ACPI 3.0 T‑states, the driver looks for support for ACPI 1.0 T‑states.

#### ACPI 3.0 T‑state Initialization

During initialization, the processor driver looks for and evaluates the \_PTC, \_TSS, and \_TSD ACPI objects. If the processor driver finds these objects, it performs validation checks to ensure that these objects are correctly populated and compatible with Windows T‑state support. The following validation checks are performed:

* \_PTC object validation
* The \_PTC must use either I/O mapped or FFH controls.
* \_TSS object validation
* The \_TSS entries must be ordered in descending order of T-state and power.
* The first \_TSS entry must be 100 percent (no throttle).
* \_TSD object validation
* The \_TSD object, if present, must describe only one coordination domain.
* The \_TSD object must contain no more than five fields.
* The \_TSD coordination type must be one of the three coordination types that are defined by ACPI 3.0.
* The \_TSD domain must contain at least one member, and the member count must not be greater than the total number of processors that are declared in the ACPI namespace.

#### ACPI 1.0 T‑state Initialization

If support for ACPI 3.0 T‑states is not present in the system firmware, the processor driver attempts to initialize ACPI 1.0 T‑states. The following validation checks are performed:

* T-states on multiprocessor systems
* The system must be a uniprocessor system. ACPI 1.0 T-states are not supported on multiprocessor systems, unless the processor driver for the system’s specific processor explicitly supports and enables the multiprocessor T‑state capability that is based on known capabilities of the system.
* Processor Register Block address
* The Processor Register Block (P\_BLK) address must be located in I/O space.
* Throttling values
* The FADT DUTY\_WIDTH value must be >0 and ≤4.
* The sum of the FADT DUTY\_OFFSET and DUTY\_WIDTH values must be ≤32.

# Core Parking

Core Parking is a new Windows kernel power manager and kernel scheduler technology that helps improve the energy efficiency of a system by dynamically scaling the number of logical processors that are in use based on workload. Similar to how processor performance states help scale the performance of a single processor, Core Parking is designed to help scale the performance and energy efficiency across the set of logical processors in the system. When the Core Parking algorithm reduces the number of logical processors that are in use, it *parks* some of the logical processors in the system. The kernel scheduler correspondingly gives preference to *unparked* logical processors rather than *parked* logical processors when it schedules any non-affinitized threads. This lets the parked logical processors become idle, which in turn lets the corresponding processor cores transition into a lower power idle state.

Core Parking is most effective on systems that have processor idle states with extremely low power consumption. When combined with ITTD, Core Parking helps reduce the amount of interrupt activity on systems that run Hyper-V. On systems with processors that include Intel Hyper-Threading Technology, Core Parking is also leveraged to help intelligently schedule work between threads that are running on multiple logical processors in the same processor core.

We have engaged with leading processor vendors to tune the Core Parking policy defaults. System manufacturers are encouraged to change the Core Parking policy parameters only with detailed consultation from the system processor vendor.

Core Parking is supported only on Windows Server 2008 R2. However, the Core Parking algorithm and infrastructure is also used to balance processor performance between logical processors on Windows 7 client systems with processors that include Intel Hyper-Threading Technology.

# Processor Clocking Control

Processor Clocking Control (PCC) is a new feature for Windows Server 2008 R2 where the operating system and the underlying platform hardware cooperatively manage PPM. With PCC, Windows continues to direct the processor performance according to the system workload and communicates the ideal amount of processor performance to another hardware entity in the system through a firmware-described interface.

Typically, systems that use PCC are server platforms that have a separate management controller that is responsible for directing processor performance and power consumption. This controller is ultimately responsible for transitioning the processor to the specific performance state or throttle state that provides the requested level of processor performance. PCC allows the system manufacturer to innovate in related power management features because the underlying hardware platform can be made aware of specific operating system processor performance requirements.

When PCC is enabled, Windows is not in direct control of the processor’s performance state, but instead communicates its performance state preference to the underlying platform hardware. Typically, the platform should provide at least the requested amount of processor performance for the given interval. However, if the system is running under a power consumption budget, thermal condition, or other extraneous condition, the platform can temporarily violate the agreement and provide reduced performance. In this situation, the platform must notify Windows about the reduction in performance. The ability of the underlying platform hardware to notify Windows about power budgets, thermal conditions, or other extraneous conditions is a key benefit of PCC. It lets Windows adjust the processor performance algorithm and notify the administrator or user that the system processor performance is currently reduced.

PCC is supported only on Windows Server 2008 R2. It is not supported on Windows 7 client systems. PCC requires an out-of-band management device to change the processor performance and throttle states and is designed solely for server systems.

# Processor Power Management Validation Tools

Microsoft provides several software tools to help you test and verify system PPM support. These tools allow you to view, change, or record the system’s use of PPM technologies. The following is a list of these tools and their general capabilities:

* Windows Resource Monitor

This toollets you view a quick summary of the processor usage versus the processor speed. This tool is included with Windows.

* Windows Reliability and Performance Monitor

This toollets you view a plot of the current speed of each processor core against the processor usage. It also provides logging and graphical display capabilities. This tool is included with Windows.

* Event Viewer

This tool lets you view entries in the system event log. The system event logincludes entries that contain summary information about platform PPM capabilities and error messages that were logged when ACPI processor objects failed validation checks. This tool is included with Windows.

* PowerCfg

This command-line tool lets you view or change the processor power policy, including power policy values that are not exposed in the Control Panel Power Options application. This tool is included with Windows.

* Pwrtest

This command-line tool lets you view detailed information about system PPM capabilities and provides test and logging capabilities for multiple power management scenarios. This tool is included with the Windows Driver Kit (WDK).

The following sections provide examples of how you can use these tools.

## Windows Resource Monitor

The Windows Resource Monitor can display a summary view of resource usage in the system, including the overall CPU usage and the use of processor throttling controls such as processor performance states and linear stop clock throttle states. This tool is shown in Figure 2.



**Frequency**

**(Average P-State)**

Figure 2. Windows Resource Monitor showing performance state usage

## Windows Reliability and Performance Monitor

The Windows Reliability and Performance Monitor can display both processor performance state and processor idle state usage.

### Viewing Processor Performance State Usage

The Windows Reliability and Performance Monitor includes performance counters that can provide a somewhat more detailed view of the processor speed changes that occur in response to changes in power policy or changes in processor utilization. By using these performance counters, you can see the current processor speed for each processor or core in the system. Figure 3 shows how to add these performance counters in the Windows Reliability and Performance Monitor.



Figure 3. Adding processor performance counters in the Reliability and Performance Monitor

Figure 4 shows an example of Windows Reliability and Performance Monitor displaying processor performance state usage.



Figure 4. Windows Reliability and Performance Monitor displaying processor performance state usage

### Viewing Processor Idle State Usage

The Windows Reliability and Performance Monitor can also be used to view and log processor idle state (APCI C‑state) usage. It can be useful to use the histogram view in the Windows Reliability and Performance Monitor to compare the system idleness to the three possible C‑states on a system. Figure 5 shows an example of Windows Reliability and Performance Monitor displaying such a histogram.



Figure 5. Windows Reliability and Performance Monitor displaying processor idle state (C‑State) usage

Although Windows 7 and Windows Server 2008 R2 use only three discrete C-states, the underlying system can use lower power processor idle states than C3 for the third C-state. System manufacturers that would like to view how much time a system spends in these lower power idle states are encouraged to contact their processor vendor for vendor-specific tools that can be used for this purpose.

## Event Viewer

The Windows kernel power manager publishes events to the system event log. At system initialization, the kernel power manager logs an informational event that provides a summary of the processor idle states that the system supports. Figure 6 shows an example of such an event when it is viewed with Event Viewer.



Figure 6. System Event Log entry that shows the supported PPM states

The kernel power manager also logs error events if any ACPI processor objects fail validation checks during processor driver initialization that cause any PPM capabilities to be disabled. Figure 7 shows two examples of such error events when viewed with Event Viewer.





Figure 7. System Event Log entries that show PPM error events

## PowerCfg

The PowerCfg command-line tool can be used to view or change the processor power policy. For information about how to use the PowerCfg tool, see “Power Policy Configuration and Deployment” on the WHDC Web site.

## Pwrtest

The Pwrtest command-line tool can be used to view the supported PPM states and to view and log processor idle state transitions. The Pwrtest tool is not included with Windows. However, it is included with the WDK.

### Viewing Supported PPM States

Pwrtest can be used to view a summary of the PPM states that the system supports by specifying the **/info:ppm** command-line option. Figure 8 shows an example of the output of Pwrtest when it is run with **/info:ppm** on a system with an Intel processor.

C:\Program Files\Microsoft PwrTest>pwrtest /info:ppm

PROCESSOR\_POWER\_INFORMATION

 CPU Number = 0

 MaxMhz = 1833

 CurrentMhz = 1833

 MhzLimit = 1833

 MaxIdleState = 3

 CurrentIdleState = 3

InstanceName: ACPI\GenuineIntel\_-\_x86\_Family\_6\_Model\_14\\_0\_0

Processor Performance States

 PerfStates:

 Max Transition Latency: 10 us

 Number of States: 11

 State Speed (Mhz) Type

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

 0 1833 (100%) Performance

 1 1333 ( 72%) Performance

 2 1000 ( 54%) Performance

 3 1000 ( 54%) Throttle

 4 880 ( 48%) Throttle

 5 750 ( 40%) Throttle

 6 630 ( 34%) Throttle

 7 500 ( 27%) Throttle

 8 380 ( 20%) Throttle

 9 250 ( 13%) Throttle

 10 130 ( 7%) Throttle

Figure 8. Pwrtest output that shows the supported PPM states

### Viewing and Logging Processor Idle State Transitions

Pwrtest can also be used to view and log processor idle state transitions. Pwrtest supports two basic modes of operation for these tests:

* A sampling mode that summarizes processor idle state usage over the sampling period. For this mode, you specify the **/ppm** command-line option.
* An event-driven mode that logs processor idle state transitions as they are initiated by the Windows kernel power manager. For this mode, you specify both the **/ppm** and the **/live** command-line options.

Figure 9 shows an example of the output of Pwrtest when it is run in sampling mode.

C:\Program Files\Microsoft PwrTest>pwrtest /ppm

 Elapsed Idle C1 C2 C3 P- Freq Freq Perf/

Cpu [ms] [%] [%] [%] [%] State [%] [MHz] Throttle

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

 0 5007 98 0 73 26 2 54 1000 P

 1 5007 99 0 93 6 2 54 1000 P

 0 10014 97 0 72 27 2 54 1000 P

 1 10014 97 0 91 8 2 54 1000 P

 0 15021 88 1 0 0 2 54 1000 P

 1 15021 89 1 0 0 2 54 1000 P

 0 20028 99 0 0 100 2 54 1000 P

 1 20028 98 0 0 100 2 54 1000 P

 0 25035 98 0 0 100 2 54 1000 P

 1 25035 98 0 0 100 2 54 1000 P

 0 30042 98 0 0 100 2 54 1000 P

 1 30042 100 0 0 100 2 54 1000 P

 0 35049 98 0 0 100 2 54 1000 P

 1 35049 98 0 0 100 2 54 1000 P

 0 40056 99 0 0 100 2 54 1000 P

 1 40056 97 0 0 100 2 54 1000 P

Figure 9. Pwrtest output that shows sampling PPM states

Figure 10 shows an example of the output of Pwrtest when it is run in event-driven mode.

C:\Program Files\Microsoft PwrTest>pwrtest /ppm /live

Waiting for PPM Events. Press 'Ctrl-C' to quit...

Timestamp Proc# Event Information

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

21:27:17.890 1 Idle State Demotion (Old:2, New:1, Affinity:0x2)

21:27:17.937 1 Idle State Promotion (Old:1, New:2, Affinity:0x2)

21:27:19.481 0 Idle State Demotion (Old:2, New:1, Affinity:0x1)

21:27:19.497 1 Idle State Demotion (Old:2, New:1, Affinity:0x2)

21:27:19.528 0 Idle State Demotion (Old:1, New:0, Affinity:0x1)

21:27:19.590 0 Idle State Promotion (Old:0, New:1, Affinity:0x1)

21:27:19.606 1 Idle State Promotion (Old:1, New:2, Affinity:0x2)

21:27:19.637 0 Idle State Promotion (Old:1, New:2, Affinity:0x1)

21:27:38.871 0 Idle State Demotion (Old:2, New:1, Affinity:0x1)

21:27:38.918 0 Idle State Promotion (Old:1, New:2, Affinity:0x1)

21:27:41.133 0 Idle State Demotion (Old:2, New:1, Affinity:0x1)

21:27:41.133 1 Idle State Demotion (Old:2, New:1, Affinity:0x2)

21:27:41.196 1 Perf State Change (State:0, Speed:1833 Mhz)

21:27:41.196 1 Domain Perf State Change

 (State:0, Speed:1833 Mhz, Affinity:0x3)

21:27:41.196 0 Idle State Demotion (Old:1, New:0, Affinity:0x1)

21:27:41.242 1 Idle State Promotion (Old:1, New:2, Affinity:0x2)

21:27:41.242 0 Idle State Promotion (Old:0, New:1, Affinity:0x1)

21:27:41.305 0 Perf State Change (State:0, Speed:1833 Mhz)

21:27:41.305 0 Idle State Demotion (Old:1, New:0, Affinity:0x1)

21:27:41.367 1 Idle State Demotion (Old:2, New:1, Affinity:0x2)

21:27:41.414 1 Idle State Promotion (Old:1, New:2, Affinity:0x2)

21:27:41.461 0 Idle State Promotion (Old:0, New:1, Affinity:0x1)

21:27:41.493 0 Idle State Demotion (Old:1, New:0, Affinity:0x1)

21:27:41.546 0 Idle State Promotion (Old:0, New:1, Affinity:0x1)

21:27:41.579 1 Idle State Demotion (Old:2, New:1, Affinity:0x2)

21:27:41.606 0 Idle State Demotion (Old:1, New:0, Affinity:0x1)

21:27:41.630 1 Idle State Demotion (Old:1, New:0, Affinity:0x2)

21:27:41.790 1 Idle State Promotion (Old:0, New:1, Affinity:0x2)

21:27:41.814 0 Idle State Promotion (Old:0, New:1, Affinity:0x1)

21:27:41.890 1 Idle State Promotion (Old:1, New:2, Affinity:0x2)

21:27:41.915 0 Idle State Promotion (Old:1, New:2, Affinity:0x1)

21:27:41.994 1 Perf State Change (State:1, Speed:1333 Mhz)

21:27:41.994 1 Domain Perf State Change

 (State:1, Speed:1333 Mhz, Affinity:0x3)

21:27:42.126 0 Perf State Change (State:1, Speed:1333 Mhz)

21:27:42.344 1 Perf State Change (State:2, Speed:1000 Mhz)

21:27:42.344 1 Domain Perf State Change

Figure 10. Pwrtest Output that shows event-driven logging of PPM states

Pwrtest provides several other command-line options that can be used to tailor the sampling rate and information that is logged in these modes. For more information, see the documentation that is included with Pwrtest and the Pwrtest command-line Help.

# Processor Power Policy on Windows

Windows 7 and Windows Server 2008 R2 include a rich set of power policy settings that can influence and control the Windows kernel power manager algorithms for choosing target PPM states. These power policy settings are not intended to be changed by end users and are not exposed in the Control Panel Power Options application. Although most systems do not require adjustment to these settings, they are provided to let system manufacturers and system administrators optimize the Windows PPM behavior.

For example, PPM policy settings can be used to do the following:

* Adapt Window kernel power manager use of PPM controls to the specific PPM features and characteristics of processors from different manufacturers.
* Account for the differences in PPM implementations or its intended application across various system roles or target workloads.
* Fine-tune Windows kernel power manager use of PPM features to account for specific system implementations, such as chipsets, system devices, and interconnects.
* Enable OEMs to offer power policies that are tuned for their specific systems and customer or market segment needs.

This section describes these PPM policy settings and their effect on Windows kernel power manager behavior and use of PPM features.

## Usage Guidelines

Most Windows 7 systems do not require changes to the PPM policy settings. However, systems that have unique processor and chipset combinations or systems that include devices that have compatibility issues with the default PPM policies might require changes to some of the PPM policy settings.

These policies are intended for static use only in Windows power plans. That is, the policy settings can be set for specific power plans, for use on both AC and DC power, and take effect when the user changes the active Windows power plan or changes the power source from AC power to battery power. Microsoft does not support any use of PPM policies other than for static configuration through Windows power plans.

These policy settings should not be exposed to the user in custom power control software. OEMs are encouraged to refrain from creating such software and should limit the use of these policy settings only to default power plan customization. Changing these policy settings to an incorrect combination of values can negatively impact system energy efficiency, performance, and responsiveness.

## Firmware Implementation of Processor Power Policy

We have observed systems that control processor performance policy from the system firmware. Specifically, the system firmware can use processor-specific registers to throttle the performance of the system without notifying Windows. Similarly, we have observed systems that constantly limit the maximum processor performance state when the system is running on battery power.

We encourage system manufacturers to leverage Windows power policy to enforce processor performance state limits. Implementing processor policy in the system firmware instead of leveraging Windows power policy can cause performance and responsiveness issues that users might not be able to diagnose. Additionally, some users require maximum system performance when running their computers on battery power and are not aware that the underlying system firmware might be limiting the processor performance. Implementing default processor performance policy by using Windows power policy lets users override the OEM defaults in Power Options in Control Panel. This can help reduce OEM support issues with users who try to obtain maximum performance at the expense of battery life.

## Windows 7 Power Policy Store

To use the processor policy settings that are described in this section, it is important to first understand how power policies are defined and stored in Windows 7.

The following fundamental rules apply:

* Individual power settings for the system or for devices are grouped into *power plans.* A power plan is a group of power setting values that help achieve a given power mode or overall power preference. The default power plans for Windows 7 are *High Performance*, *Balanced*, and *Power Saver*.
* Every power setting belongs to a logical subgroup of related settings. Some examples of these subgroups include system sleep settings, display settings, and PPM settings.
* Each power setting has both an AC and a DC value. The AC value is used when the system is plugged in (running on AC power). The DC value is used with the system is powered by a battery (running on DC power).
* Every power setting is identified by a unique globally unique identifier (GUID).
* Changing a power setting simply changes that value in the underlying power policy store. For the updated value to take effect on the system, a power plan that uses the new power setting value must be applied to the system.

The most significant change in Windows 7 compared to Windows Vista in relation to the power settings is that in Windows 7 the PPM policies are all discrete power setting values in the power policy store. In Windows Vista, the processor power settings were grouped into collections of settings such as Performance State settings (*ppmperf*) or Idle State settings (*ppmidle*). In Windows 7, each power setting is an individual top-level setting within the PPM settings subgroup.

For example, to change the value of the Processor Performance Time Check Interval setting on Windows Vista, the current time check interval value must first be retrieved by running the PowerCfg tool with the **/ppmidle** and **/decode** command-line options. Then, the value for the setting can be updated, re-encoded (by using the **/encode** command-line option), and applied to the system. However, for Windows 7, the Processor Performance Time Check Interval setting is its own setting in the PPM settings subgroup and its value can be updated directly by running the PowerCfg tool with the **/setacvalueindex** or **/setdcvalueindex** command-line option.

For more information about how to use the PowerCfg tool, see “Power Policy Configuration and Deployment” on the WHDC Web site.

## Processor Power Policy Settings

This section provides details on the processor power policy settings that are available in Windows 7 to influence kernel power manager use of PPM. All the following settings are part of the Processor Power Management Settings (SUB\_PROCESSOR) subgroup.

### Processor Performance Policy Settings

These policy settings configure the processor performance state (P-state) algorithm. Except for the Processor Performance Minimum Processor Stateand Processor Performance Maximum Processor State settings, these policy settings should not be changed without consultation from the system processor vendor or manufacturer. Changes to PPM policy settings require extensive validation for power savings and system performance.

Processor Performance Increase Threshold

|  |  |
| --- | --- |
| Description | The percentage of processor utilization, in terms of maximum processor utilization, that is required to increase the processor to a higher performance state. |
| GUID | 06cadf0e-64ed-448a-8927-ce7bf90eb35d |
| PowerCfg Alias | Not applicable |
| Minimum Value | 0 |
| Maximum Value | 100 |
| Label | Percentage (%) |
| Hidden | Yes |

Processor Performance Decrease Threshold

|  |  |
| --- | --- |
| Description | The percentage of processor utilization, in terms of maximum processor utilization, that is required to reduce the processor to a lower performance state. |
| GUID | 12a0ab44-fe28-4fa9-b3bd-4b64f44960a6 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 0 |
| Maximum Value | 100 |
| Label | Percentage (%) |
| Hidden | Yes |

Allow Throttle States

|  |  |
| --- | --- |
| Description | Specifies that the performance state algorithm can use any ACPI throttle states (T-states) that the system supports. |
| GUID | 3b04d4fd-1cc7-4f23-ab1c-d1337819c4bb |
| PowerCfg Alias | Not applicable |
| Possible Values | Index | Name | Description |
| 0 | Disabled | Throttle states are not used. |
| 1 | Enabled | Throttle states can be used. However, the processor throttle state does not change adaptively. When enabled, the Minimum Processor State and Maximum Processor Statesettings can be used to lock the system processors into a specific processor throttle state. |
| Hidden | Yes |

Processor Performance Decrease Policy

|  |  |
| --- | --- |
| Description | Specifies how a target performance state is selected if the current processor utilization is below the value of the Processor Performance Decrease Thresholdsetting. |
| GUID | 465e1f50-b610-473a-ab58-00d1077dc418 |
| PowerCfg Alias | Not applicable |
| Possible Values | Index | Name | Description |
| 0 | Ideal | A target performance state is selected based on calculating which performance state increases the processor utilization to just above the value of the Processor Performance Decrease Thresholdsetting*.* |
| 1 | Single | The next lowest performance state (compared to the current performance state) is selected. |
| 2 | Rocket | The lowest performance state is selected. |
| Hidden | Yes |

Processor Performance Increase Policy

|  |  |
| --- | --- |
| Description | Specifies how a target performance state is selected if the current processor utilization is above the value of the Processor Performance Increase Threshold setting. |
| GUID | 40fbefc7-2e9d-4d25-a185-0cfd8574bac6 |
| PowerCfg Alias | Not applicable |
| Possible Values | Index | Name | Description |
| 0 | Ideal | A target performance state is selected based on calculating which performance state decreases the processor utilization to just below the value of the Processor Performance Increase Thresholdsetting. |
| 1 | Single | The next highest performance state (compared to the current performance state) is selected. |
| 2 | Rocket | The highest performance state is selected. |
| Hidden | Yes |

Processor Performance Time Check Interval

|  |  |
| --- | --- |
| Description | Specifies the duration, in milliseconds, between subsequent evaluations of the processor performance state and Core Parking algorithms. |
| GUID | 4d2b0152-7d5c-498b-88e2-34345392a2c5 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 1 |
| Maximum Value | 5000 |
| Label | Milliseconds (ms) |
| Hidden | Yes |

Processor Performance History Count

|  |  |
| --- | --- |
| Description | Specifies the number of Processor Performance Time Check Intervalsover which a processor’s performance history is tracked. |
| GUID | 7d24baa7-0b84-480f-840c-1b0743c00f5f |
| PowerCfg Alias | Not applicable |
| Minimum Value | 1 |
| Maximum Value | 128 |
| Label | Time Check Intervals |
| Hidden | Yes |

Minimum Processor State

|  |  |
| --- | --- |
| Description | Specifies the minimum processor performance state. The performance state is specified as a percentage of maximum processor frequency. |
| GUID | 893dee8e-2bef-41e0-89c6-b55d0929964c |
| PowerCfg Alias | PROCTHROTTLEMIN |
| Minimum Value | 0 |
| Maximum Value | 100 |
| Label | Percentage (%) |
| Hidden | No |

Maximum Processor State

|  |  |
| --- | --- |
| Description | Specifies the maximum processor performance state. The performance state is specified as a percentage of maximum processor frequency. |
| GUID | bc5038f7-23e0-4960-96da-33abaf5935ec |
| PowerCfg Alias | PROCTHROTTLEMAX |
| Minimum Value | 0 |
| Maximum Value | 100 |
| Label | Percentage (%) |
| Hidden | No |

Processor Performance Decrease Time

|  |  |
| --- | --- |
| Description | Specifies the minimum amount of time that must elapse between subsequent reductions in the processor performance state. The time is specified in units of the number of Processor Performance Time Check Intervals. |
| GUID | d8edeb9b-95cf-4f95-a73c-b061973693c8 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 1 |
| Maximum Value | 100 |
| Label | Time Check Intervals |
| Hidden | Yes |

Processor Performance Increase Time

|  |  |
| --- | --- |
| Description | Specifies the minimum amount of time that must elapse between subsequent increases in the processor performance state. The time is specified in units of the number of Processor Performance Time Check Intervals. |
| GUID | 984cf492-3bed-4488-a8f9-4286c97bf5aa |
| PowerCfg Alias | Not applicable |
| Minimum Value | 1 |
| Maximum Value | 100 |
| Label | Time Check Intervals |
| Hidden | Yes |

Processor Performance Boost Policy

|  |  |
| --- | --- |
| Description | Configures the processor performance boost policy. The behavior of this setting can differ between processor vendors and specific processor models. The processor vendor should be consulted before changing the value of this setting. |
| GUID | 45bcc044-d885-43e2-8605-ee0ec6e96b59 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 0 |
| Maximum Value | 100 |
| Label | Percentage (%) |
| Hidden | Yes |

### Processor Core Parking Policy Settings

These policy settings configure the Core Parking algorithm parameters. These policy settings should not be changed without consultation from the system processor vendor.

Processor Performance Core Parking Increase Threshold

|  |  |
| --- | --- |
| Description | The percentage of processor utilization (in terms of maximum processor utilization across all unparked logical processors in the system) that is required to transition additional logical processors from the parked state to the unparked state. |
| GUID | df142941-20f3-4edf-9a4a-9c83d3d717d1 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 5 |
| Maximum Value | 90 |
| Label | Percentage (%) |
| Hidden | Yes |

Processor Performance Core Parking Decrease Threshold

|  |  |
| --- | --- |
| Description | The percentage of processor utilization (in terms of maximum processor utilization across all unparked logical processors in the system) that is required to transition additional logical processors from the unparked state to the parked state. |
| GUID | 68dd2f27-a4ce-4e11-8487-3794e4135dfa |
| PowerCfg Alias | Not applicable |
| Minimum Value | 5 |
| Maximum Value | 90 |
| Label | Percentage (%) |
| Hidden | Yes |

Processor Performance Core Parking Overutilization Threshold

|  |  |
| --- | --- |
| Description | The percentage of processor utilization (in terms of maximum processor utilization for a single logical processor) above which a given logical processor is determined to be overutilized. An overutilized logical processor is a logical processor that is in the parked state with processor utilization that is greater than the value of this setting. This occurs when the logical processor is running sufficient affinitized work. This setting helps the Core Parking algorithm identify which particular logical processors to transition from the parked state to the unparked state. |
| GUID | 943c8cb6-6f93-4227-ad87-e9a3feec08d1 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 5 |
| Maximum Value | 100 |
| Label | Percentage (%) |
| Hidden | Yes |

Processor Performance Core Parking Increase Policy

|  |  |
| --- | --- |
| Description | Specifies the number of logical processors to transition from the parked state to the unparked state when the processor utilization across all unparked logical processors in the system exceeds the value of the Processor Performance Core Parking Increase Threshold setting. |
| GUID | c7be0679-2817-4d69-9d02-519a537ed0c6 |
| PowerCfg Alias | Not applicable |
| Possible Values | Index | Name | Description |
| 0 | Ideal | As many logical processors as are required to reduce the processor utilization below the value of the Processor Performance Core Parking Increase Thresholdsetting are transitioned from the parked state to the unparked state. |
| 1 | Single | A single logical processor is transitioned from the parked state to the unparked state. On the next evaluation of the Core Parking algorithm, if the processor utilization is still greater than the value of the Processor Performance Core Parking Increase Threshold setting, another single logical processor is transitioned from the parkedstate to the unparked state. |
| 2 | All (Rocket) | All logical processors that are in the parkedstate are transitioned to the unparkedstate. Note that the value of the Processor Performance Core Parking Maximum Coressetting can limit the number of logical processors that can be in the unparked state at the same time. |
| Hidden | Yes |

Processor Performance Core Parking Decrease Policy

|  |  |
| --- | --- |
| Description | Specifies the number of logical processors to transition from the unparkedstate to the parked state when the processor utilization across all unparked logical processors in the system is below the value of the Processor Performance Core Parking Decrease Threshold setting. |
| GUID | 71021b41-c749-4d21-be74-a00f335d582b |
| PowerCfg Alias | Not applicable |
| Possible Values | Index | Name | Description |
| 0 | Ideal | As many logical processors as are required to increase the processor utilization above the value of the Processor Performance Core Parking Decrease Threshold setting are transitioned from the unparkedstate to the parked state. |
| 1 | Single | A single processor is transitioned from the unparkedstate to the parked state. On the next evaluation of the Core Parking algorithm, if the processor utilization is still less than the value of the Processor Performance Core Parking Decrease Threshold setting, another single processor is transitioned from the unparkedstate to the parked state. |
| 2 | All (Rocket) | All processor cores that are in the unparkedstate that can be parked are transitioned to the parkedstate. Note that at least one logical processor per node always remains in the unparked state. Note that the value of the Processor Performance Core Parking Minimum Coressetting can increase the number of logical processors that always remain in the unparked state. |
| Hidden | Yes |

Processor Performance Core Parking Max Cores

|  |  |
| --- | --- |
| Description | The maximum percentage of logical processors (in terms of all logical processors that are enabled on the system) that can be in the unparked state at any given time. For example, on a system with 16 logical processors, configuring the value of this setting to 50% ensures that no more than 8 logical processors are ever in the unparked state at the same time. The Core Parking algorithm is disabled if the value of this setting is not greater than the value of the Processor Performance Core Parking Minimum Coressetting. |
| GUID | ea062031-0e34-4ff1-9b6d-eb1059334028 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 0 |
| Maximum Value | 100 |
| Label | Percentage (%) |
| Hidden | Yes |

Processor Performance Core Parking Min Cores

|  |  |
| --- | --- |
| Description | The minimum percentage of logical processors (in terms of all logical processors that are enabled on the system) that can be placed in the unparked state at any given time. For example, on a system with 16 logical processors, configuring the value of this setting to 25% ensures that at least 4 logical processors are always in the unparked state. The Core Parking algorithm is disabled if the value of this setting is not less than the value of the Processor Performance Core Parking Maximum Coressetting. |
| GUID | 0cc5b647-c1df-4637-891a-dec35c318583 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 0 |
| Maximum Value | 100 |
| Label | Percentage (%) |
| Hidden | Yes |

Processor Performance Core Parking Increase Time

|  |  |
| --- | --- |
| Description | Specifies the minimum amount of time that must elapse before additional logical processors can be transitioned from the parked state to the unparked state. The time is specified in units of the number of Processor Performance Time Check Intervals. |
| GUID | 2ddd5a84-5a71-437e-912a-db0b8c788732 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 1 |
| Maximum Value | 100 |
| Label | Time Check Intervals |
| Hidden | Yes |

Processor Performance Core Parking Decrease Time

|  |  |
| --- | --- |
| Description | Specifies the minimum amount of time that must elapse before additional logical processors can be transitioned from the unparked state to the parkedstate. The time is specified in units of the number of Processor Performance Time Check Intervals. |
| GUID | dfd10d17-d5eb-45dd-877a-9a34ddd15c82 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 1 |
| Maximum Value | 100 |
| Label | Time Check Intervals |
| Hidden | Yes |

The Windows kernel scheduler notifies the Core Parking algorithm when affinitized threads are run on logical processors that are currently in the parked state. This helps the Core Parking algorithm select the logical processors that have recently been scheduled affinitized work to transition from the parked state to the unparked state. The following policy settings configure the Core Parking algorithm for maintaining and tracking instances of affinitized work that have been scheduled to parked logical processors.

Processor Performance Core Parking Affinity History Decrease Factor

|  |  |
| --- | --- |
| Description | Specifies the factor by which to decrease the affinity history for each logical processor. The rate is specified in units of the number of Processor Performance Time Check Intervals. |
| GUID | 8f7b45e3-c393-480a-878c-f67ac3d07082 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 1 |
| Maximum Value | 10 |
| Label | None (discrete units) |
| Hidden | Yes |

Processor Performance Core Parking Affinity History Threshold

|  |  |
| --- | --- |
| Description | Specifies the affinity history value above which a logical processor is considered to have recently been scheduled significant affinitized work. |
| GUID | 5b33697b-e89d-4d38-aa46-9e7dfb7cd2f9 |
| PowerCfg Alias | Not applicable. |
| Minimum Value | 1 |
| Maximum Value | 1000 |
| Label | None (discrete units) |
| Hidden | Yes |

Processor Performance Core Parking Affinity Weighting

|  |  |
| --- | --- |
| Description | Specifies the weighting that is given to each occurrence of affinitized work that is scheduled to a logical processor. |
| GUID | e70867f1-fa2f-4f4e-aea1-4d8a0ba23b20 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 0 |
| Maximum Value | 1000 |
| Label | None (discrete units) |
| Hidden | Yes |

The Core Parking algorithm tracks logical processors that are overutilized. An overutilized logical processor is a logical processor that is in the parked state that had processor utilization during the last processor performance time check interval that is greater than the value of the Processor Performance Core Parking Overutilization Threshold setting. This helps the Core Parking algorithm select the particular parked logical processors that have recently been scheduled significant amounts of work to transition to the unparked state.

Processor Performance Core Parking Overutilization History Decrease Factor

|  |  |
| --- | --- |
| Description | Specifies the factor by which to decrease the overutilization history for each logical processor. The rate is specified in units of the number of Processor Performance Time Check Intervals. |
| GUID | 1299023c-bc28-4f0a-81ec-d3295a8d815d |
| PowerCfg Alias | Not applicable |
| Minimum Value | 1 |
| Maximum Value | 10 |
| Label | None (discrete units) |
| Hidden | Yes |

Processor Performance Core Parking Overutilization History Threshold

|  |  |
| --- | --- |
| Description | Specifies the overutilization history value above which a logical processor is considered to have recently been overutilized. |
| GUID | 9ac18e92-aa3c-4e27-b307-01ae37307129 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 1 |
| Maximum Value | 1000 |
| Label | None (discrete units) |
| Hidden | Yes |

Processor Performance Core Parking Overutilization Weighting

|  |  |
| --- | --- |
| Description | Specifies the weighting that is given to each occurrence where a logical processor that is in the parked state is determined to be overutilized. |
| GUID | 8809c2d8-b155-42d4-bcda-0d345651b1db |
| PowerCfg Alias | Not applicable |
| Minimum Value | 0 |
| Maximum Value | 1000 |
| Label | None (discrete units) |
| Hidden | Yes |

Processor Performance Core Parking Core Override

|  |  |
| --- | --- |
| Description | Specifies that at least one logical processor per processor core must always remain in the unparked state regardless of the values of the Processor Performance Core Parking Maximum Cores and Processor Performance Core Parking Minimum Coressettings. |
| GUID | a55612aa-f624-42c6-a443-7397d064c04f |
| PowerCfg Alias | n/a |
| Possible Values | Index | Name | Description |
| 0 | Disabled | Allow all logical processors in a processor core to be parked. |
| 1 | Enabled | Require at least one logical processor per processor core to remain unparked. |
| Hidden | Yes |

Processor Performance Core Parking Parked Performance State

|  |  |
| --- | --- |
| Description | Specifies the performance state (P-state) for a logical processor that is transitioned from the unparked state to the parked state. |
| GUID | 447235c7-6a8d-4cc0-8e24-9eaf70b96e2b |
| PowerCfg Alias | Not applicable |
| Possible Values | Index | Name | Description |
| 0 | No Preference | The current performance state for the logical processor is maintained. |
| 1 | Lowest Performance State | The logical processor will be put into the lowest supported (Pmax) performance state. |
| 2 | Highest Performance State | The logical processor will be put into the highest supported (P0) performance state. |
| Hidden | Yes |

Note that if a logical processor is a member of a processor performance state domain that includes other logical processors, the actual performance state of the logical processor is set to the highest performance state of all logical processors in the processor performance state domain.

### Processor Idle Policy Settings

These policy settings configure the processor idle state (C-state) algorithm. These policy settings should not be changed without consultation from the system processor and chipset vendor.

Processor Idle Time Check

|  |  |
| --- | --- |
| Description | Specifies the duration, in microseconds, between subsequent evaluations of the processor idle state algorithm.  |
| GUID | c4581c31-89ab-4597-8e2b-9c9cab440e6b |
| PowerCfg Alias | Not applicable |
| Minimum Value | 1 |
| Maximum Value | 200000 |
| Label | Microseconds (us) |
| Hidden | Yes |

Processor Idle Demote Threshold

|  |  |
| --- | --- |
| Description | The amount of processor idleness that is required before a processor is set to the next higher power processor idle state. When the processor idleness goes below the value of this setting, the processor transitions to the next lower numbered C-state. |
| GUID | 4b92d758-5a24-4851-a470-815d78aee119 |
| PowerCfg Alias | Not applicable |
| Minimum Value | 0 |
| Maximum Value | 100 |
| Label | Percentage (%) |
| Hidden | Yes |

Processor Idle Promote Threshold

|  |  |
| --- | --- |
| Description | The amount of processor idleness that is required before a processor is set to the next lower power processor idle state. When the processor idleness goes above the value of this setting, the processor transitions to the next higher numbered C-state. |
| GUID | 7b224883-b3cc-4d79-819f-8374152cbe7c |
| PowerCfg Alias | Not applicable |
| Minimum Value | 0 |
| Maximum Value | 100 |
| Label | Percentage (%) |
| Hidden | Yes |

Processor Idle Disable

|  |  |
| --- | --- |
| Description | Specifies if processor idle states are disabled on the system. Note that disabling the processor idle states on a system has severe negative energy efficiency and thermal consequences and should be done only in consultation with the system or processor vendor. |
| GUID | 5d76a2ca-e8c0-402f-a133-2158492d58ad |
| PowerCfg Alias | Not applicable |
| Possible Values | Index | Name | Description |
| 0 | Enabled | Processor idle states are enabled. |
| 1 | Disabled | Processor idle states are disabled. |
| Hidden | Yes |

Processor Idle Threshold Scaling

|  |  |
| --- | --- |
| Description | Specifies if the processor idleness is scaled to the current processor performance state when calculating a target processor idle state. |
| GUID | 6c2993b0-8f48-481f-bcc6-00dd2742aa06 |
| PowerCfg Alias | Not applicable |
| Possible Values | Index | Name | Description |
| 0 | Disabled | Idle threshold scaling is disabled. |
| 1 | Enabled | Idle threshold scaling is enabled. |
| Hidden | Yes |

### System Cooling Policy Setting

This policy setting configures how Windows responds to high thermal conditions on systems that support active cooling features, such as fans.

System Cooling Policy

|  |  |
| --- | --- |
| Description | Configures Windows thermal behavior on systems that support active cooling features. |
| GUID | 94d3a615-a889-4ac5-ae2b-e4d8f634367f |
| PowerCfg Alias | Not applicable |
| Possible Values | Index | Name | Description |
| 0 | Passive | The system reduces the processor performance before it enables active cooling features such as fans. |
| 1 | Active | The system enables active cooling features such as fans before it reduces the processor performance. |
| Hidden | No |

# Next Steps

System designers, firmware developers, and OEMs are encouraged to:

* Carefully implement the ACPI processor objects in the system firmware on capable systems as previously described in this paper to ensure proper support for PPM when running Windows.
* Leverage the Pwrtest tool in the WDK and the tools included with Windows, including PowerCfg, to help you test and validate PPM support in Windows 7 and Windows Server 2008 R2.

# Resources

ACPI / Power Management - Architecture and Driver Support

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

ACPI in Windows Vista (WinHEC 2006)

<http://download.microsoft.com/download/5/b/9/5b97017b-e28a-4bae-ba48-174cf47d23cd/CPA002_WH06.ppt>

ACPI Specification – Revision 3.0b

<http://www.acpi.info/>

Extended PSS ACPI Method Specification

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

Processor Power Management in Windows Vista and Windows Server 2008

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

Power Optimizations for Windows Platforms (WinHEC 2007)

<http://download.microsoft.com/download/a/f/d/afdfd50d-6eb9-425e-84e1-b4085a80e34e/SYS-T302_WH07.pptx>

Power Policy Configuration and Deployment in Windows

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

Using PowerCfg to Evaluate System Energy Efficiency

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

WDK and Developer Tools

<http://www.microsoft.com/whdc/DevTools/default.mspx>