Windows Certification Program

Hardware Certification Taxonomy & Requirements

December 2014

Microsoft Corporation Technical Documentation License Agreement

READ THIS! THIS IS A LEGAL AGREEMENT BETWEEN MICROSOFT CORPORATION ("MICROSOFT") AND THE RECIPIENT OF THESE MATERIALS, WHETHER AN INDIVIDUAL OR AN ENTITY ("YOU"). IF YOU HAVE ACCESSED THIS AGREEMENT IN THE PROCESS OF DOWNLOADING MATERIALS ("MATERIALS") FROM A MICROSOFT WEB SITE, BY CLICKING "I ACCEPT", DOWNLOADING, USING OR PROVIDING FEEDBACK ON THE MATERIALS, YOU AGREE TO THESE TERMS. IF THIS AGREEMENT IS ATTACHED TO MATERIALS, BY ACCESSING, USING OR PROVIDING FEEDBACK ON THE ATTACHED MATERIALS, YOU AGREE TO THESE TERMS.

For good and valuable consideration, the receipt and sufficiency of which are acknowledged, You and Microsoft agree as follows:

1. You may review these Materials only (a) as a reference to assist You in planning and designing Your product, service or technology ("Product") to interface with a Microsoft Product as described in these Materials; and (b) to provide feedback on these Materials to Microsoft. All other rights are retained by Microsoft; this agreement does not give You rights under any Microsoft patents. You may not (i) remove this agreement or any notices from these Materials, or (ii) give any part of these Materials, or assign or otherwise provide Your rights under this agreement, to anyone else.

2. These Materials may contain preliminary information or inaccuracies, and may not correctly represent any associated Microsoft Product as commercially released. All Materials are provided entirely "AS IS." To the extent permitted by law, MICROSOFT MAKES NO WARRANTY OF ANY KIND, DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, AND ASSUMES NO LIABILITY TO YOU FOR ANY DAMAGES OF ANY TYPE IN CONNECTION WITH THESE MATERIALS OR ANY INTELLECTUAL PROPERTY IN THEM.

3. If You are an entity and (a) merge into another entity or (b) a controlling ownership interest in You changes, Your right to use these Materials automatically terminates and You must destroy them.

4. You have no obligation to give Microsoft any suggestions, comments or other feedback ("Feedback") relating to these Materials. However, any Feedback you voluntarily provide may be used in Microsoft Products and related specifications or other documentation (collectively, "Microsoft Offerings") which in turn may be relied upon by other third parties to develop their own Products. Accordingly, if You do give Microsoft Feedback on any version of these Materials or the Microsoft Offerings to which they apply, You agree: (a) Microsoft may freely use, reproduce, license, distribute, and otherwise commercialize Your Feedback in any Microsoft Offering; (b) You also grant third parties, without charge, only those patent rights necessary to enable other Products to use or interface with any specific parts of a Microsoft Product that incorporate Your Feedback; and (c) You will not give Microsoft any Feedback (i) that You have reason to believe is subject to any patent, copyright or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any Microsoft Offering incorporating or derived from such Feedback, or other Microsoft intellectual property, to be licensed to or otherwise shared with any third party.

5. Microsoft has no obligation to maintain confidentiality of any Microsoft Offering, but otherwise the confidentiality of Your Feedback, including Your identity as the source of such Feedback, is governed by Your NDA.

6. This agreement is governed by the laws of the State of Washington. Any dispute involving it must be brought in the federal or state superior courts located in King County, Washington, and You waive any defenses allowing the dispute to be litigated elsewhere. If there is litigation, the losing party must pay the other party's reasonable attorneys' fees, costs and other expenses. If any part of this agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall continue in effect. This agreement is the entire agreement between You and Microsoft concerning these Materials; it may be changed only by a written document signed by both You and Microsoft.

Release Notes

This publication of the Windows Hardware Certification Program Requirements provides an update to the Windows 8 and Windows 8.1 Certification Program. These requirement changes are intended to relax the Windows 8.1 system and device requirements and give our partners greater flexibility in designing and differentiating their products in 2015.

It is important to understand the changes are to remove or modify the specific requirements listed under Summary of Changes only. All other requirements will remain to support device interoperability, compatibility with Windows, and application platform consistency. The tests associated with these removed or modified requirements will remain in the HCK to aid in your testing and measurement of your system's quality. A set of HCK filters will be provided for the purposes of achieving a passing result needed for certification.

Summary of Changes

The following changes are made in this updated document.

Requirement	Change Type	Summary of Changes
System.Fundamentals.TPM20.TPM20 Required	Removed	Microsoft Windows will not require TPM 2.0 on Windows 8 and Windows 8.1 systems. TPM 2.0 is required for connected standby systems, and if implemented on non-Connected Standby systems. The text of the requirement moves to optional engineering requirements in System.Fundamentals.TPM20.TPM20.
System.Fundamentals.TPM20.TPM20	Modified	Moved text from removed requirement System.Fundamentals.TPM20.TPM2.0Required to this one, and marked it as optional.
System.Fundamentals.TPM20.EKcerts	Modified	Changed title to make it if implemented. We are not changing the enforcement date as the requirement is no longer required, it is optional.

The following changes have been made in April 2014.

Requirement	Change Type	Summary of Changes
System.Client.PowerManagement.BatteryLife	Removed	Requirement removed to allow for greater flexibility in system design.
System.Client.PowerManagement.CaseTemperature	Removed	Requirement removed as it is no longer relevant for current system design.
System.Client.SystemConfiguration.Windows8RequiredComponents	Modified	Requirement modified to adjust required internal storage free space.
System.Client.Webcam.VideoCaptureAndCamera	Modified	Requirement modified to allow for greater flexibility in system design.
System.Fundamentals.DebugPort.SystemExposesDebugInterface	Modified	Requirement modified to provide additional detail on separate debugging boards.
System.Fundamentals.StorageAndBoot.BootPerformance	Modified	Requirement modified to update eMMC version and to adjust for related free space changes.

System Requirements

System.Client.Aero

Desktop Windows Manager (DWM) is the desktop graphical user interface system that enables the Windows Aero user interface and visual theme. This feature is required to be enabled for all Windows 7 systems, but the memory bandwidth requirement applies to certain form factors.

Related Requirements	• System.Client.Aero.SystemsStartAeroTheme

System.Client.Aero.SystemsStartAeroTheme

Systems are capable of starting the Aero theme

Target Feature	System.Client.Aero
Applies to	• Windows 7 Client x86, x64

Description

All Windows systems must be capable of starting the Aero theme, by meeting the following requirements. All Windows systems must enable the Aero theme by default when running with a version of Windows that includes Aero.

Requirements:

- All video hardware in all system form factors must support a minimum 32 bpp.
- All systems must meet the following DirectX requirements:
    - All Windows 7 systems are required to either support the D3De9X interface and be Direct3D 9.x compliant, OR support Direct3D 10 or greater.
    - All Windows 7 systems except ultra-mobile form factor PCs are required to include graphics hardware that supports Direct3D 10 or greater.
    - All Windows 7 ultra-mobile form factor PCs are required to either support the D3De9X interface and be Direct3D 9.x compliant, or support Direct3D 10 or greater.
- All systems must meet the following minimum video memory size requirements.
    - For desktop that ship with a monitor and have one video port connector, all-in-one systems that do not have a video port for a second monitor, and for all mobile and ultra mobile systems, the following amount of total graphics memory (either discrete and/or total non-local) is required for specified monitor resolutions. Monitor resolutions are expressed as total pixels (X dimension multiplied by Y dimension):
        - Resolution less than 1,310,720 (1280x1024) pixels include 64MB.
        - Resolutions equal to or greater than 1,310,720 (1280x1024) pixels and equal to or less than 2,304,000 (1920x1200) pixels include 128MB.
        - Resolutions greater than 2,304,000 (1920x1200) pixels include 256MB.

- For desktop systems that do not ship with a monitor, a minimum of 128MB of total graphics memory (either discrete and/or total non-local) memory is required.
- For all dual monitor capable desktop systems that ship with a monitor (systems with two physical and operational monitor connections)
    - Resolution less than 2,621,440 (1280x1024 by 2) pixels includes 128MB.
    - Resolutions equal to or greater than 2,621,440 (1280x1024 by 2) pixels includes 256MB.
- For dual monitor capable desktop systems that do not ship with a monitor, 256MB of total graphics memory (either discrete and/or total non-local) memory is required.
- Mobile and ultra mobile systems with dual monitor capabilities are excluded from the dual monitor desktop requirements and are treated as single monitor systems. This is due to the different usage models and that external monitor ports are often used in mirroring or "duplicate" mode.

- Systems must meet the following minimum video memory performance requirements, as measured by WinSAT:
    - Desktop systems must have a measured memory bandwidth of 1,600 megabytes per second (MB/s) at a monitor resolution of 1,310,720 (equivalent to 1280x1024).
    - Mobile systems and all-in-one designs must achieve a measured bandwidth of 1,600 megabytes per second (MB/s) measured at the native resolution of the shipping panel.
    - Ultra mobile systems have no minimum video memory performance requirement.

Design Notes:
Memory requirements described here are for usable or active connectors. Microsoft does not recommend supporting fewer active connections than available connectors on a given system as this causes failed expectations and poor user experience. If a system supports m connectors, but limits the number of active connections to n (where n < m), then memory requirements are applicable for n connectors as long as there is no way for a user to activate greater than n connections at any time.

Additional Information

Enforcement Date	Jun. 01, 2007

System.Client.BluetoothController.Base

These requirements apply to systems that have generic Bluetooth controllers

Related Requirements	• System.Client.BluetoothController.Base.4LeSpecification • System.Client.BluetoothController.Base.CS • System.Client.BluetoothController.Base.LEStateCombinations • System.Client.BluetoothController.Base.LEWhiteList • System.Client.BluetoothController.Base.NoBluetoothLEFilterDriver • System.Client.BluetoothController.Base.OnOffStateControllableViaSoftware • System.Client.BluetoothController.Base.RadioScanIntervalSettings • System.Client.BluetoothController.Base.SimultaneousBrEdrAndLeTraffic • System.Client.BluetoothController.Base.SystemsWithBluetoothImplementDeviceID • System.Client.BluetoothController.Base.WLANBTCoexistence

System.Client.BluetoothController.Base.4LeSpecification

If a system includes a Bluetooth controller it must support the Bluetooth 4.0 specification requirements

Target Feature	System.Client.BluetoothController.Base
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The Bluetooth controller must comply with the Basic Rate (BR) and Low Energy (LE) Combined Core Configuration Controller Parts and Host/Controller Interface (HCI) Core Configuration requirements outlined in the Compliance Bluetooth Version 4.0 specifications.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.BluetoothController.Base.CS

Systems that support Connected Standby with Bluetooth controllers must ship with Microsoft's inbox Bluetooth stack

Target Feature	System.Client.BluetoothController.Base
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Systems that support Connected Standby that ship with Bluetooth controllers must ship with Microsoft's inbox Bluetooth stack.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.BluetoothController.Base.LEStateCombinations

Systems with Bluetooth Controllers must support a minimum set of LE state combinations

Target Feature	System.Client.BluetoothController.Base

Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The Bluetooth controller must allow the spec LE state combinations (as allowed in section [Vol 6] Part B, Section 1.1.1 of the Bluetooth version 4.0 spec).

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.BluetoothController.Base.LEWhiteList

Systems with Bluetooth controllers must support a minimum LE white list size of 25 entries

Target Feature	System.Client.BluetoothController.Base
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The Bluetooth controller on the System must support a minimum of 25 entries in its white list for remote Low Energy (LE) devices.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.BluetoothController.Base.NoBluetoothLEFilterDriver

Bluetooth LE filter drivers are not allowed to load on BTHLEENUM.SYS

Target Feature	System.Client.BluetoothController.Base
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

To ensure a uniform experience across Windows Store Apps using the Bluetooth LE (GATT) WinRT API, filter drivers shall not be loaded on BTHLEENUM.SYS.

Additional Information

Business Justification	The GATT WinRT API provided for communication over Bluetooth LE is closely coupled to the driver implementing GATT support for the inbox Bluetooth stack, BTHLEENUM.SYS. All Windows Store Apps that use the Microsoft WinRT API for GATT rely on this interface to be work as originally implemented, thus there shall not be any 3rd party components that may intentionally or inadvertently affect this interface.
Enforcement Date	Jun. 26, 2013

System.Client.BluetoothController.Base.OnOffStateControllableViaSoftware

Bluetooth controllers' On/Off state must be controllable via software

Target Feature	System.Client.BluetoothController.Base
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

For Certifying Bluetooth controllers for Windows 8.1:

When turning the radio "off", Bluetooth controllers shall be powered down to its lowest supported power state and no transmission/reception shall take place. Windows will terminate Bluetooth activity by unloading the inbox protocol drivers and their children, submitting the HCI_Reset command to the controller, and then setting the controller to the D3 logical power state, allowing bus drivers to power down the radio as appropriate. The radio can be completely powered off if a bus-supported method is available to turn the radio back on. No additional vendor software control components will be supported.

On turning the radio back on, the Windows Bluetooth stack shall resume the device to D0, allowing bus drivers to restart the device. The Windows Bluetooth stack shall then reinitialize the Bluetooth components of the controller.

Bluetooth Radio Management in Windows 8.1 shall only be enabled for internal Bluetooth 4.0 controllers.

For Windows 8 Certified controllers on upgrade to Windows 8.1:

On upgrade to Windows 8.1, previous DLL support for Bluetooth 4.0 controllers shall be ignored and the Bluetooth controller is expected to be, at a minimum, in a state where there is no transmission/reception from the antenna.

For Certifying Bluetooth controllers for Windows 8 only:

The previous requirement remains unchanged.

Bluetooth controllers' On/Off state shall be controllable via software as described in Bluetooth Software Radio Switch The Off state is defined, at a minimum, as disabling the antenna component of the Bluetooth module so there can be no transmission/reception. There must not be any hardware-only switches to control power to the Bluetooth radio.

The radio must maintain on/off state across sleep and reboot.

Additional Information

Business Justification	The Windows 8.1 implementation of Bluetooth Radio Management provides for an improved Radio Management experience while decreasing the work needed by our IHV and OEM partners.
Enforcement Date	Jun. 26, 2013

System.Client.BluetoothController.Base.RadioScanIntervalSettings

Bluetooth controllers must use radio scan interval values as set by Windows

Target Feature	System.Client.BluetoothController.Base
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

To ensure a uniform experience that balances power usage with responsiveness for users in reconnect scenarios, Bluetooth controllers must use the Page Scan Interval and LE Scan Interval values as set by Windows at all times.

Additional Information

Business Justification	To ensure a uniform experience that balances power usage with responsiveness for users in reconnect scenarios, Bluetooth controllers must use the Page Scan Interval and LE Scan Interval values as set by Windows at all times.
Enforcement Date	Jun. 26, 2013

System.Client.BluetoothController.Base.SimultaneousBrEdrAndLeTraffic

Bluetooth controllers must support simultaneous BR/EDR and LE traffic.

Target Feature	System.Client.BluetoothController.Base
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Bluetooth controllers must allow the simultaneous use of both Basic Rate (BR)/Enhanced Data Rate (EDR) and Low Energy (LE) radios.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.BluetoothController.Base.SystemsWithBluetoothImplementDeviceID

Systems which support Bluetooth must implement the DeviceID profile, version 1.3

Target Feature	System.Client.BluetoothController.Base
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Systems which support Bluetooth must include the Device ID record as specified in the DeviceID profile, version 1.3. This record shall contain the device's VID/PID.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.BluetoothController.Base.WLANBTCoexistence

Windows Systems that support both WLAN and Bluetooth must meet WLAN-BT Co-existence requirements.

Target Feature	System.Client.BluetoothController.Base
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Windows Systems that support both WLAN and Bluetooth must meet WLAN-BT Co-existence requirements listed below. The requirement is applicable to all WLAN devices across all bus types.

- Must meet 700kbps throughput on BT averaged over 2 minutes while connected to WLAN in ExTSTA mode.
- Must not drop the connection with WLAN AP when BT is scanning for new devices.
- Must be able to scan simultaneously for both WLAN and BT networks.
- Must not regress the WLAN throughput expectations listed in WLAN device performance requirement (Device.Network.WLAN.Base.MeetPerformanceReq) when BT is connected or is in scanning mode.
- Must support all WLAN concurrency combinations listed in WLAN device concurrency requirement (Device.Network.WLAN.WiFiDirect.SupportAtLeast2WiFiDirectPortsConcurrently) when BT is connected or is in scanning mode.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.BluetoothController.NonUSB

These requirements apply to systems that have non-USB Bluetooth controllers

Related Requirements	• System.Client.BluetoothController.NonUSB.NonUsbUsesMicrosoftsStack • System.Client.BluetoothController.NonUSB.ScoSupport

System.Client.BluetoothController.NonUSB.NonUsbUsesMicrosoftsStack

Any platform using a non-USB connected BT controller must ship with MSFT's inbox BT stack

Target Feature	System.Client.BluetoothController.NonUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Any platform using a non-USB connected BT controller must ship with MSFT's inbox BT stack

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.BluetoothController.NonUSB.ScoSupport

Any platform with a non-USB connected Bluetooth controller must use a sideband channel for SCO

Target Feature	System.Client.BluetoothController.NonUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Any platform using a Non-USB connected Bluetooth controller must use sideband channel for SCO (eg. SCO over an I^2S/PCM interface)

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.BluetoothController.USB

These requirements apply to systems that have USB Bluetooth controllers

Related Requirements	• System.Client.BluetoothController.USB.ScoDataTransportLayer

System.Client.BluetoothController.USB.ScoDataTransportLayer

Bluetooth host controllers support the SCO data transport layer as specified in the Bluetooth 2.1+EDR specifications

Target Feature	System.Client.BluetoothController.USB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

A System with a Bluetooth controller must comply with the Synchronous Connection Oriented (SCO)-USB requirements that are outlined in the Specification of the Bluetooth System, Version 2.1 + Enhanced Data Rate (EDR), Part A, Section 3.5.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Client.BrightnessControls

This section describes requirements systems with brightness controls.

Related Requirements	• System.Client.BrightnessControls.BacklightOptimization • System.Client.BrightnessControls.BrightnessControlButtons • System.Client.BrightnessControls.SmoothBrightness

System.Client.BrightnessControls.BacklightOptimization

WDDM 1.2 drivers must enable scenario based backlight power optimization to reduce backlight level used by integrated panel .

Target Feature	System.Client.BrightnessControls
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

- If WDDM driver supports scenario based backlight power optimization, it must indicate the support by implementing the DXGK_BRIGHTNESS_INTERFACE2 interface

- When Windows sets the current scenario by using the DxgkDdiSetBacklightOptimization function, the WDDM driver is required to honor the intent of the scenario as follows:
    - DxgkBacklightOptimizationDisable: Driver is required to completely disable all backlight optimization.
    - DxgkBacklightOptimizationDesktop: Driver is required to enable backlight optimization at a lower aggressiveness level. Driver must optimize for scenarios like photo viewing, browser, and Office documents.
    - DxgkBacklightOptimizationDynamic: Driver is required to enable backlight optimization at a higher aggressiveness level. Driver must optimize for scenarios like video playback and gaming.
    - DxgkBacklightOptimizationDimmed: Driver is required to enable backlight optimization at a higher aggressiveness level. Driver must make sure that the content on the screen is visible but it need not be easily readable.
- Driver is allowed to dynamically change the aggressiveness level based on the content on the screen
- Driver is required to handle Windows requests for change to brightness level (based on user input or ambient light sensor) while keeping backlight optimization enabled
- Driver is required to gradually transition between aggressiveness levels.
    - This is important in the case when user briefly invokes playback controls. At that time, Windows will reset the scenario from DxgkBacklightOptimizationDynamic to DxgkBacklightOptimizationDesktop. The transition must not be a step but must be gradual.
- WDDM driver is required to provide accurate information when Windows queries DxgkDdiGetBacklightReduction
- Connecting additional display devices to the system must not impact the ability to perform backlight optimization on the integrated panel of the system.

Additional Information

Business Justification	Windows 8 is optimized for thin and light, touch based tablet devices. One of the key attributes of such a device is a long battery life. On a device like this, one of the largest consumers of power is the display backlight. The amount of power consumed is directly proportional to the brightness level produced by the backlight. The Windows hardware ecosystem has innovated in this area and designed algorithms to consume less battery power by optimizing the backlight level in certain scenarios without having a significant negative impact on the user experience. This can be achieved by enhancing the colors of individual pixels to compensate for the lower backlight level of the display. There are some scenarios where such an optimization can be enabled without having a significant negative impact on the user experience. The key characteristics of such a scenario are a series of rapidly changing frames. This is typically seen in scenarios like video playback. Other possibilities are game play. However, there is one challenge with this. Windows 8 has been designed to provide a visually rich user experience. Enabling a power saving optimization like this

	during a scenario that requires a rich visual experience could have an undesired impact on the user experience. In particular there are some scenarios which require the best visual experience. For example, image viewing, Start page, graphics design, etc. So this means that we need to establish a scenario based policy on whether or not to enable such optimizations. The policy would need to balance the rich visual experience against the desired battery life. This is called scenario based backlight power optimization. This feature allows a graphics vendor to enable such optimization at the right times and save batter power, without having significant impact on the user experience.
Enforcement Date	Jun. 26, 2013

System.Client.BrightnessControls.BrightnessControlButtons

Mobile systems that have brightness control function keys use standard ACPI events and support control of LCD backlight brightness via ACPI methods in the system firmware

Target Feature	System.Client.BrightnessControls
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The Windows® Mobility Center provides users with an LCD brightness control user interface. If the system implements keys that are invisible to the operating system, these keys must use Advanced Configuration and Power Interface (ACPI) methods. These keys must not directly control the brightness after Bit 2 of the _DOS method has been set. This requires the implementation of ACPI brightness methods in the system firmware.

The following methods are required:

- _BCL
- _BCM

Bit 2 of the _DOS method must be disabled so that the system firmware cannot change the brightness levels automatically.

The following methods are optional:

Support for the _BQC method is highly recommended but not required. Systems must map keys to the following ACPI notification values:

- ACPI_NOTIFY_CYCLE_BRIGHTNESS_HOTKEY 0x85
- ACPI_NOTIFY_INC_BRIGHTNESS_HOTKEY 0x86
- ACPI_NOTIFY_DEC_BRIGHTNESS_HOTKEY 0x87
- ACPI_NOTIFY_ZERO_BRIGHTNESS_HOTKEY 0x88

Design Notes:

The _BCL and _BCM methods in the firmware enable the operating system to query the brightness range and values and to set new values. Refer to the ACPI 3.0 specification for more details.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.BrightnessControls.SmoothBrightness

Driver must support a smooth transition in response to all brightness change requests from Windows

Target Feature	System.Client.BrightnessControls
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

1. All Windows 8 systems that support brightness control, are required to support smooth brightness control
2. All Windows 8 systems are required to report 101 brightness levels to Windows. Brightness is reported as a % so this means 0 to 100 levels, including 0 and 100. Internally the driver might support more granular brightness control.
3. This is to ensure that Windows has the ability to make fine grained changes to the screen brightness. However, the brightness slider UI might expose fewer levels through the slider because it might be cumbersome for the user to adjust so many levels.
4. WDDM driver is required to implement smooth brightness control in the driver without depending on the embedded controller (EC) for the smoothness.
5. WDDM driver is required to indicate support for smooth brightness control using the capability bit defined in the DXGK_BRIGHTNESS_INTERFACE2 interface.
6. WDDM driver must enable/disable smooth brightness control based on state set using DxgkDdiSetBrightnessState.
7. When Windows requests a change to brightness, driver is required to gradually change the brightness level over time so that the change is not a step.
8. WDDM driver is allowed to select an appropriate slope for transition. However, the transition must complete in less than 2s.
9. WDDM driver is allowed to alter the slope based on panel characteristics to ensure smoothness of brightness control.
10. WDDM driver is required to start responding immediately to new brightness level requests. This must be honored even if the system is already in the process of an existing transition. At such a time, the system must stop the existing transition at the current level and start the new transition from the current position. This will ensure that when a user is using the slider to manually adjust the brightness, the brightness control is still responsive and not sluggish.

11. WDDM driver is required to continue supporting smooth brightness control, even if content based adaptive brightness optimization is currently in effect.
12. When WDDM driver is pnp started, it must detect the brightness level applied by the firmware and smoothly transition from that level to the level set by Windows.
13. Connecting additional display devices to the system must not impact the ability to do smooth brightness control on the integrated panel of the system.
14. Brightness levels are represented as a % in Windows. Therefore there is no absolute mapping between brightness % level and physical brightness level. For Windows 8, the following is the guidance

Percent represented to Windows	User Experience
0%	Brightness level such that the contents of the screen are barely visible to the user
100%	Max brightness supported by panel

Additional Information

Business Justification	Windows 8 is designed to provide a fluid user experience. There are many scenarios in which the brightness level of the display is changed. This could be based on user input, user action or based on some sensors. In all these cases, a sudden change to the brightness level is jarring. Therefore it is valuable to provide a smooth brightness transition to ensure a good user experience.
Enforcement Date	Aug. 01, 2012

System.Client.Buttons

This section refers to requirements related to buttons.

Related Requirements	• System.Client.Buttons.WindowsButtons

System.Client.Buttons.WindowsButtons

All in one systems that can run on a battery must have a Windows button

Target Feature	System.Client.Buttons
Applies to	• Windows 8.1 Client x86, x64

Description

The Windows button must have the Windows glyph as outlined in the Logo License Agreement. The Windows button size must be large enough for the Windows Logo artwork as well as the required white space as defined in the Logo License Agreement. The button can be any shape (for example: circular, square, or rectangular). The Windows button can be placed anywhere on the system. Microsoft prefers that the button be integrated centered along the bottom bezel, but the integration is at the choice of the OEM. It can be integrated on any bezel or edge of the system.

Additional Information

Business Justification	These hardware buttons enable a navigation experience in the absence of a keyboard. System builders should consider whether any type of dynamic feedback is required for any capacitive buttons for visually impaired users.
Enforcement Date	Jun. 26, 2013

System.Client.CPU

This section describes requirements for WOA SoC CPUs

Related Requirements	• System.Client.CPU.Compatibility • System.Client.CPU.MADT

System.Client.CPU.Compatibility

System contains a Windows-8 compatible System on Chip Applications Processor

Target Feature	System.Client.CPU
Applies to	• Windows 8 Client ARM (Windows RT) • Windows 8.1 Client ARM (Windows RT 8.1)

Description

Details

The following are required.

Item	Processor Attribute
Architecture	ARMv7-A application profile with MP extensions
Cores	2 or more
ISA	Thumb-2 ISA
Virtualization	Not required
Security Extensions	Not required
Interrupt Controller	GIC
RTC	Any RTC with appropriate UEFI and ACPI support which conforms to Windows 8 requirements //PlugAndPlay.WakeAlarmDevice The minimum requirements are to get and set time, get and set a wake alarm.
Debugging	ARMv7-A CP14 debug coprocessor and CP15 performance monitors infrastructure Memory-mapped debugging is not supported.
DMA	For each system DMA controller, all channels in the controller must be able to service all devices (request

	lines) connected to that controller. All devices using DMA must support the maximum physical address space of the CPU. We recommend bus mastering for devices using DMA.
Caches	Physically Indexed Physically Tagged (PIPT) Data Cache PIPT and Virtually Indexed Physically Tagged (VIPT) Instruction Cache
Performance Counter	Invariant per-processor counter (recommended) or Invariant platform counter Minimum frequency is 1 MHz Minimum rollover is 4 seconds
Clock Timer	Per processor invariant timer (recommended) or an invariant platform timer Minimum frequency is 1 KHz
Always On Timer and counter	Invariant platform counter and timer, they must remain on at the lowest SoC power state
Profiling Timer	Per processor timer
Scaling Source	A timer with known frequency with less than 100 PPM drift or Accurate EfiStall EFI boot service
Processor Clock And Power Gating	The processor can enter halt state The processor can be clock gated The processor can be power gated Processor Dynamic Frequency and Voltage Scaling (DFVS)
SIMD and VFP	NEON and VFPv3 D32

Windows will support only the ARMv7-A application profile, as described in the *ARM Architecture Reference Manual ARMv7-A Edition* (http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0406b/index.html).

Comments

- A counter is a readable register that can be used to measure elapsed time between reads, whereas a timer is capable of generating an interrupt after a predetermined interval. They can both be implemented in a single block or separately. The separation in this document is based on how they are used by the OS, and doesn't necessarily imply implementation requirements.
- Accurate EFI stall is a stall implementation that synchronizes to the edge of a fixed frequency clock, such that the additional delay introduced on top of any requested stall, is bounded by one clock cycle of that fixed frequency clock.
- The requirement for multiple cores is based on implementations available at this writing. High performance single-core processors may be approved on a case-by-case basis.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.CPU.MADT

ARM System must implement MP Start Protocol

Target Feature	System.Client.CPU

Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Details

- A system with an ARM processor must correctly implement the MP start protocol.
- The Multiple APIC Description Table (MADT) which is detailed in ACPI indicates the number of processors in the system. The number of processors mentioned in MADT must match the number of processors that Windows has enumerated using GetSystemInfo API.
- Details of the MP start protocol are available on request and will eventually be posted on MSDN.
- Details referenced above are currently in Appendix B of Minimum UEFI Requirements for SoC Platforms ver. 0.92.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Base

Base feature for digitizer

Related Requirements	• System.Client.Digitizer.Base.DigitizersAppearAsHID • System.Client.Digitizer.Base.HighQualityDigitizerInput

System.Client.Digitizer.Base.DigitizersAppearAsHID

Please refer to Device.Digitizer.DigitizersAppearAsHID

Target Feature	System.Client.Digitizer.Base
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Base.DigitizersAppearAsHID**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Base.HighQualityDigitizerInput

Please refer to Device.Digitizer.Base.HighQualityDigitizerInput

Target Feature	System.Client.Digitizer.Base
Applies to	• Windows 7 Client x86, x64

Description

Please refer to **Device.Digitizer.Base.HighQualityDigitizerInput**

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.Digitizer.Pen

Pen feature for digitizer

Related Requirements	• System.Client.Digitizer.Pen.100HzSampleRate • System.Client.Digitizer.Pen.ContactAccuracy • System.Client.Digitizer.Pen.HoverAccuracy • System.Client.Digitizer.Pen.PenRange • System.Client.Digitizer.Pen.PenResolution

System.Client.Digitizer.Pen.100HzSampleRate

Please refer to Device.Digitizer.Pen.100HzSampleRate

Target Feature	System.Client.Digitizer.Pen
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Pen.100HzSampleRate**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Pen.ContactAccuracy

Please refer to Device.Digitizer.Pen.ContactAccuracy

Target Feature	System.Client.Digitizer.Pen
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Pen.ContactAccuracy**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Pen.HoverAccuracy

Please refer to Device.Digitizer.Pen.HoverAccuracy

Target Feature	System.Client.Digitizer.Pen
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Pen.HoverAccuracy**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Pen.PenRange

Please refer to Device.Digitizer.Pen.PenRange

Target Feature	System.Client.Digitizer.Pen
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Pen.PenRange**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Pen.PenResolution

Please refer to Device.Digitizer.Pen.PenResolution

Target Feature	System.Client.Digitizer.Pen
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Pen.PenResolution**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch

Windows Touch interface for digitizer devices.

Related Requirements	• System.Client.Digitizer.Touch.5TouchPointMinimum • System.Client.Digitizer.Touch.DigitizerConnectsOverUSBOrI2C • System.Client.Digitizer.Touch.DigitizerJitter • System.Client.Digitizer.Touch.ExtraInputBehavior • System.Client.Digitizer.Touch.FieldFirmwareUpdatable • System.Client.Digitizer.Touch.HIDCompliantFirmware • System.Client.Digitizer.Touch.HighQualityTouchDigitizerInput • System.Client.Digitizer.Touch.HighResolutionTimeStamp • System.Client.Digitizer.Touch.InputSeparation • System.Client.Digitizer.Touch.NoiseSuppression • System.Client.Digitizer.Touch.PhysicalDimension • System.Client.Digitizer.Touch.PhysicalInputPosition • System.Client.Digitizer.Touch.PowerStates • System.Client.Digitizer.Touch.ReportingRate • System.Client.Digitizer.Touch.ResponseLatency • System.Client.Digitizer.Touch.TouchResolution • System.Client.Digitizer.Touch.ZAxisAllowance

System.Client.Digitizer.Touch.5TouchPointMinimum

Please refer to Device.Digitizer.Touch.5TouchPointMinimum

Target Feature	System.Client.Digitizer.Touch

Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.5TouchPointMinimum**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.DigitizerConnectsOverUSBOrI2C

Please refer to Device.Digitizer.Touch.DigitizerConnectsOverUSBOrI2C

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.DigitizerConnectsOverUSBOrI2C**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.DigitizerJitter

Please refer to Device.Digitizer.Touch.DigitizerJitter

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.DigitizerJitter**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.ExtraInputBehavior

Please refer to Device.Digitizer.Touch.ExtraInputBehavior

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.ExtraInputBehavior**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.FieldFirmwareUpdatable

Please refer to Device.Digitizer.Touch.FieldFirmwareUpdatable

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.FieldFirmwareUpdatable**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.HIDCompliantFirmware

Please refer to Device.Digitizer.Touch.HIDCompliantFirmware

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.HIDCompliantFirmware**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.HighQualityTouchDigitizerInput

Please refer to Device.Digitizer.Touch.HighQualityTouchDigitizerInput

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 7 Client x86, x64

Description

Please refer to **Device.Digitizer.Touch.HighQualityTouchDigitizerInput**

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.Digitizer.Touch.HighResolutionTimeStamp

Please refer to Device.Digitizer.Touch.HighResolutionTimeStamp

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.HighResolutionTimeStamp**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.InputSeparation

Please refer to Device.Digitizer.Touch.InputSeparation

Target Feature	System.Client.Digitizer.Touch

Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.InputSeparation**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.NoiseSuppression

Please refer to Device.Digitizer.Touch.NoiseSuppression

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.NoiseSuppression**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.PhysicalDimension

Please refer to Device.Digitizer.Touch.PhysicalDimension

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.PhysicalDimension**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.PhysicalInputPosition

Please refer to Device.Digitizer.Touch.PhysicalInputPosition

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.PhysicalInputPosition**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.PowerStates

Please refer to Device.Digitizer.Touch.PowerStates

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.PowerStates**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.ReportingRate

Please refer to Device.Digitizer.Touch.ReportingRate

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.ReportingRate**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.ResponseLatency

Please refer to Device.Digitizer.Touch.ResponseLatency

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.ResponseLatency**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.TouchResolution

Please refer to Device.Digitizer.Touch.TouchResolution

Target Feature	System.Client.Digitizer.Touch
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.TouchResolution**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Digitizer.Touch.ZAxisAllowance

Please refer to Device.Digitizer.Touch.ZAxisAllowance

Target Feature	System.Client.Digitizer.Touch

Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Please refer to **Device.Digitizer.Touch.ZAxisAllowance**

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Firewall

The requirements in this section describe what is required.

Related Requirements	• System.Client.Firewall.FirewallEnabled

System.Client.Firewall.FirewallEnabled

Systems enable a firewall solution by default

Target Feature	System.Client.Firewall
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

All systems must test and ship with firewall software solution enabled by default.

Design Notes:
The firewall can either be the inbox solution found under Security Center or a third party equivalent

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Firmware.UEFI.GOP

This section describes requirements for systems implementing UEFI firmware.

Related Requirements	• System.Client.Firmware.UEFI.GOP.Display

System.Client.Firmware.UEFI.GOP.Display

System firmware must support GOP and Windows display requirements

Target Feature	System.Client.Firmware.UEFI.GOP
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Every firmware on a Windows 8 client system must support the Graphics Output Protocol (GOP) as defined in UEFI 2.3.1.

The display is controlled by the system UEFI before the WDDM graphics driver takes over. GOP must be available when the Windows EFI boot manager loads. VBIOS is not supported. It is also required for prior UI, such as OEM logo, firmware setup, or password prompt screens to enable GOP. During this time when the firmware is in control, the following are the requirements:

Topology Selection

- UEFI must reliably detect all the displays that are connected to the POST adapter. The Pre-OS screen can only be displayed on a display connected to the POST adapter.
- In case multiple displays are detected, UEFI must display the Pre-OS screen based on the following logic
  - System with an Integrated display(Laptop, All In One, Tablet): UEFI must display the Pre-OS screen only on the integrated display
  - System **without** an Integrated display (integrated display is shut or desktop system): UEFI must display the Pre-OS screen on one display. UEFI must select the display by prioritizing the displays based on connector type. The prioritization is as follows: DisplayPort, HDMI, DVI, HD15, Component, S-Video. If there are multiple monitors connected using the same connector type, the firmware can select which one to use.

Mode Selection

- Once UEFI has determined which display to enabled to display the Pre-OS screen, it must select the mode to apply based on the following logic
  - System with an Integrated display(Laptop, All In One, Tablet): The display must always be set to its native resolution and native timing
  - System **without** an Integrated display (desktop):
    - UEFI must attempt to set the native resolution and timing of the display by obtaining it from the EDID.
    - If that is not supported, UEFI must select an alternate mode that matches the same aspect ratio as the native resolution of the display.
    - At the minimum, UEFI must set a mode of 1024 x 768
    - If the display device does not provide an EDID, UEFI must set a mode of 1024 x 768
  - The firmware must always use a 32 bit linear frame buffer to display the Pre-OS screen
  - PixelsPerScanLine must be equal to the HorizontalResolution.
  - PixelFormat must be PixelBlueGreenRedReserved8BitPerColor. Note that a physical frame buffer is required; PixelBltOnly is not supported.

Mode Pruning

- UEFI must prune the list of available modes in accordance with the requirements called out in EFI_GRAPHICS_OUTPUT_PROTOCOL.QueryMode() (as specified in the UEFI specification version 2.1)

Providing the EDID

- Once the UEFI has set a mode on the appropriate display (based on Topology Selection), UEFI must obtain the EDID of the display and pass it to Windows when Windows uses the EFI_EDID_DISCOVERED_PROTOCOL (as specified in the UEFI specification version 2.1)to query for the EDID.
  - It is possible that some integrated panels might not have an EDID in the display panel itself. In this case, UEFI must manufacture the EDID. The EDID must accurately specify the native timing and the physical dimensions of the integrated panel
  - If the display is not integrated and does not have an EDID, then the UEFI does not need to manufacture an EDID

Additional Information

Business Justification	Modern boot experience requires a pre-boot environment which is both fast and visually appealing. The system UEFI controls the display before Windows takes over the control. This means that the screen controlled by the firmware is the first thing that the user sees. Therefore, it is very important that the user has a great user experience at this stage. Some of the key goals are: Ensure that the screen is visible on exactly one display. Display on a single screen ensures that it is easy for the firmware to set a timing and that the UI is not scaled to fit multiple displays of different sizes and aspect ratios. It is easier for the firmware to display on one display instead of many. The native resolution is important in a number of Windows scenarios: Native resolution provides the sharpest and most clear text. Booting the system in native resolution eliminates the need to change modes during the boot process. The frame buffer can be handed off between bios, boot loader, OS boot, and display driver. The result of this is that the display does not flash during boot and gives a more seamless boot experience. Providing the EDID to Windows is important so that Windows can determine the physical dimensions of the display. Windows will automatically scale its UI to be large on high DPI displays so that the text is large enough for the user to see.
Enforcement Date	Mar. 01, 2012

System.Client.Graphics

This section describes requirements for graphics devices in client PC systems.

Related Requirements	• System.Client.Graphics.FullGPU • System.Client.Graphics.NoMoreThanOneInternalMonitor • System.Client.Graphics.SingleGPU • System.Client.Graphics.WDDM • System.Client.Graphics.WDDMSupportRotatedModes • System.Client.Graphics.Windows7.MinimumDirectXLevel • System.Client.Graphics.WirelessUSBDisplay

System.Client.Graphics.FullGPU

A Windows client system must have a "Full" graphics device and that device must be the post device.

Target Feature	System.Client.Graphics
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

WDDM 1.2 introduces multiple driver/device types: Full, Render only, and Display only. For a detailed description of each, refer to the WDDM 1.2 specification or the Windows 8 WDDM 1.2 requirement Device.Graphics.WDDM12.Base.

Each of these driver/device types are designed for specific scenarios and usage case. All client scenarios expect a "full" graphics device. Also many applications assume that the post device is the "best" graphics devices and use that device exclusively. For this reason, a Windows client system must have a "full" graphics driver/device that is capable of display, rendering, and video.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.Graphics.NoMoreThanOneInternalMonitor

Graphics driver must not enumerate more than one monitor as internal

Target Feature	System.Client.Graphics
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The graphics driver must not enumerate more than one display target of the D3DKMDT_VOT_INTERNAL type on any adapter.

Design Notes:

For more information, see the Graphics guide for Windows 7 at http://go.microsoft.com/fwlink/?LinkId=237084.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Graphics.SingleGPU

ARM system may not include multiple GPUs

Target Feature	System.Client.Graphics
Applies to	• Windows 8 Client ARM (Windows RT) • Windows 8.1 Client ARM (Windows RT 8.1)

Description

An ARM system may not include multiple GPUs. All mode, power management and scenarios must be available through this single GPU. So-called hybrid GPU solutions are not allowed.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.Graphics.WDDM

All Windows graphics drivers must be WDDM

Target Feature	System.Client.Graphics
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

The Windows Display Driver Model (WDDM) was introduced with Windows Vista as a replacement to the Windows XP Display Driver Model (XDDM). The WDDM architecture offers functionality to enable features such as desktop composition, enhanced fault Tolerance, video memory manager, scheduler, cross process sharing of D3D surfaces and so on. WDDM was specifically designed for modern graphics devices that are a minimum of Direct3D 10 Feature Level 9_3 with pixel shader 2.0 or better and have all the necessary hardware features to support the WDDM functionality of memory management, scheduling, and fault tolerance. WDDM for Windows Vista was referred to as "WDDM v1.0". WDDM 1.0 is required for Windows Vista.

Windows 7 made incremental changes to the driver model for supporting Windows 7 features and capabilities and is referred to as "WDDM v1.1" and is a strict superset of WDDM 1.0. WDDM v1.1 introduces support for D3D11, GDI hardware acceleration, Connecting and Configuring Displays, DXVA HD, and other features. WDDM 1.1 is required for Windows 7.

Windows 8 also introduces features and capabilities that require graphics driver changes. These incremental changes range from small changes such as smooth rotation, to large changes such as 3D stereo, and D3D11 video support. The WDDM driver model that provides these Windows 8 features is referred to as "WDDM v1.2." WDDM v1.2 is a superset of WDDM 1.1, and WDDM 1.0.

WDDM v1.2 is required by all systems shipped with Windows 8. WDDM 1.0 and WDDM 1.1 will only be supported with legacy devices on legacy systems. The best experience and Windows 8 specific features are only enabled by a WDDM 1.2 driver. A WDDM driver that implements some WDDM 1.2 required features, but not all required features will fail to load on Windows 8.

For Windows 8 XDDM is officially retired and XDDM drivers will no longer load on Windows 8 Client or Server.

Windows 8.1 introduces features and capabilities that require graphic driver changes. WDDMv1.3 brings significant improvement in areas related to performance, power and reliability for Windows.

WDDMv1.3 is required by all systems shipped with Windows 8.1.

Below is a summary these WDDM versions:

Operating System	Driver Models Supported	D3D versions supported	Features enabled
Windows Vista	WDDM 1.0 XDDM on Server and limited UMPC	D3D9, D3D10	Scheduling, Memory Management, Fault tolerance, D3D9 & 10
Windows Vista SP1 / Windows 7 client pack	WDDM 1.05 XDDM on Server 2008	D3D9, D3D10, D3D10.1	+ BGRA support in D3D10, D3D 10.1
Windows 7	WDDM 1.1 XDDM on Server 2008 R2	D3D9, D3D10, D3D10.1, D3D11	GDI Hardware acceleration, Connecting and configuring Displays, DXVA HD, D3D11
Windows 8	WDDM 1.2	D3D9, D3D10, D3D10.1, D3D11, D3D11.1	Smooth Rotation, 3D Stereo, D3D11 Video, GPU Preemption, TDR Improvements Diagnostic Improvements, Performance and Memory usage Optimizations, Power Management, etc.

WDDM v1.2 also introduces new types of graphics drivers, targeting specific scenarios and is described below:

1. WDDM Full Graphics Driver: This is the full version of the WDDM graphics driver that supports hardware accelerated 2D & 3D operations. This driver is fully capable of handling all the render, display and video functions. WDDM 1.0 and WDDM 1.1 are full graphics drivers. All Windows 8 client systems must have a full graphics WDDM 1.2 device as the primary boot device.

2. WDDM Display Only Driver: This driver is only supported as a WDDM 1.2 driver and enables IHVs to write a WDDM based kernel mode driver that is capable of driving display only devices. The OS handles the 2D or 3D rendering using a software simulated GPU. Display only devices are only allowed on client systems within a virtual machine session.

3. WDDM Render Only Driver: This driver is only supported as a WDDM 1.2 driver and enables IHVs to write a WDDM driver that supports only rendering functionality. Render only devices are not allowed as the primary graphics device on client systems.

Table below explains the scenario usage for the new driver types:

	Client	Server	Client running in a Virtual Environment	Server Virtual
Full Graphics	Required as post device	Optional	Optional	Optional
Display Only	Not allowed	Optional	Optional	Optional
Render Only	Optional as non primary adapter	Optional	Optional	Optional
Headless	Not allowed	Optional	N/A	N/A

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.Graphics.WDDMSupportRotatedModes

If accelerometer is present, WDDM driver must support all rotated modes

Target Feature	System.Client.Graphics
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

On a system with an accelerometer, the WDDM driver is required to support all rotated modes for every resolution enumerated for the integrated panel.

- A WDDM driver is required to enumerate source modes for the integrated display. The WDDM driver must support rotated modes (0,90,180 and 270) for every mode that it enumerates for the integrated panel.
- The rotation is required to be supported even if the integrated panel is in a duplicate or extended topology with another display device. For duplicate mode, it is acceptable to rotate all targets connected to the rotated source. Per path rotation is allowed but not required.

Both the above mentioned requirements are optional for Stereo 3D capable resolutions.

Additional Information

Business Justification	Windows 8 is designing some key experiences that depend on the ability of a user to be able to rotate the physical device. For this experience, it is critical that the desktop also rotate to be in sync with the device. Therefore the WDDM driver must support rotation modes.
Enforcement Date	Jun. 26, 2013

System.Client.Graphics.Windows7.MinimumDirectXLevel

All system display adapters or chipset complies with Direct3D 10 and DXGI feature sets or greater

Target Feature	System.Client.Graphics
Applies to	• Windows 7 Client x86, x64

Description

For all systems, except ultra-mobile PC form-factor, all graphics hardware in the system must be at least Direct3D 10 or greater.

The graphics memory performance target for Direct3D 10 hardware is 1,600 MB/sec and will be measured by using the AeroAT tool.
All features required by this specification must also be exposed including those features defined for the DXGI DLL. The following list includes some of the required features called out in the Direct3D 10 specification:

- Geometry shader
- Stream output
- Integer instruction set
- New compressed formats
- Render to vertex buffer
- Render to cube map
- Render to volume

Additional Information

Enforcement Date	Dec. 01, 2009

System.Client.Graphics.WirelessUSBDisplay

System must not add support for non-traditional display connectors like USB and Wireless

Target Feature	System.Client.Graphics
Applies to	• Windows 8 Client ARM (Windows RT) • Windows 8.1 Client ARM (Windows RT 8.1)

Description

- Display devices (Monitor, LCD, TV, Projectors) are enumerated to Windows only via the WDDM Graphics driver.
- Such display devices must be physically connected to a full WDDM graphics hardware that supports at least DX 9.c in the hardware.
- Windows only supports a fixed set of display connectors as defined in WDDM as part of the D3DKMDT_VIDEO_OUTPUT_TECHNOLOGY enumeration.
- The WDDM driver is required to accurately report the connection medium used to connect the display device to the system.
- This enumeration does not include any support for display devices connected via USB port or a wireless connection.

Additional Information

Business Justification	Windows was designed to work with display devices physically connected to the WDDM graphics adapter and the scan out taking place from the graphics memory. However, based on the current design this is not possible for connectors like Wireless or USB. This causes challenges in the area of: Premium content protection Dynamic changes in available bandwidth (USB and Wireless bandwidth is shared with other devices like storage, networking etc. and not dedicated to display scan out)User experience related to discovery, pairing, usage It would require driver to driver communication that is not currently supported Ability to display the screen in cases where the WDDM graphics driver is not yet running (POST screen, bug check, PnP Stop)Would require proprietary conversion from USB/Wireless to standard mediums like HDMI, DVI as the standards for USB/Wireless display continue to stabilize
Enforcement Date	Jun. 01, 2006

System.Client.MediaTranscode

Requirements describe the capabilities that are required for media transcode.

Related Requirements	• System.Client.MediaTranscode.GlitchFreeRealtimeCommunication • System.Client.MediaTranscode.SystemTranscodeFasterThanRealTime

System.Client.MediaTranscode.GlitchFreeRealtimeCommunication

Glitch free real-time video communication

Target Feature	System.Client.MediaTranscode
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

1. The system shall sustain two concurrent decoding/encoding pairs, 1 H.264 hardware encode/decode at 720p plus 1 H.264/VC-1 software encode/decode at 240p, in low latency mode glitch free for 1 hour.

2. The system shall sustain four concurrent decoding/encoding pairs, consisting of hardware H.264 decoding/encoding pairs at 540p, 360p and 240p plus 1 software H.264/VC-1 encode/decode at 240p, in low latency mode glitch free for 1 hour.

3. The system shall sustain four concurrent hardware H.264 decoding/encoding pairs in low latency mode in a glitch free manner for 1 hour with dynamic codec configuration changes every 5-20 sec. The codec configuration changes include:

   a. media type changes (resolution, profile, frame rate, sample aspect ratio) and

   b. CodecAPI change (temporal layer count).

"Glitch-free" is defined as following: For encoder the time interval between IMFTransform::ProcessInput and METransformHaveOutput must be within 25ms except that one frame can be encoded within 66ms in a per-10 second span. For decoder the time interval between Execute and output surface ready must be within 33ms.

Additional Information

Business Justification	In video conference calls such as supported by Lync and Skype the clients usually receive multiple video streams and are also capable of sending multiple video streams. The reason for requiring to encode and send multiple video streams is because the multiple receivers may request different resolutions or profiles and also may be on different downlink bandwidth links. The three requirements address several scenarios: 1. The client is sending an HD video stream and a low resolution video stream to two different receiving endpoints and at the same time is receiving two video streams. This requirement will ensure that there aren't any noticeable glitches in the video 2. The second requirement covers larger video conferences where there are multiple receivers which all request different resolutions or all have very different downlink speeds and thus require the sender to send out multiple video streams (e.g. 540p, 360p, 240p). Note that on purpose the resolution 720p is not included here as the macro-block processing rate of the encoder should not be exceeded. If then yet another video stream is needed then the application can use a software encoder to generate the fourth stream. On the receiving side again the requirement is due to

	multiple videos being shown in the Lync client. 3. In video conference calls the requests of the receiving clients can change significantly during the call e.g. if the video window of the client UI changes in size and thus requires a different resolution, if the bandwidth changes significantly e.g. by adding desktop sharing modality, if client leave or join the conference. To ensure that the encoder/decoder handles these dynamic changes the third requirement will test dynamic configuration changes.
Enforcement Date	Jun. 26, 2013

System.Client.MediaTranscode.SystemTranscodeFasterThanRealTime

System is capable of transcoding faster than real time for multimedia scenarios, both on AC and DC power

Target Feature	System.Client.MediaTranscode
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Required

Successfully transcode a Standard Definition (SD) video to another SD video from and to the following formats:

- MPEG2-PS SD to H264 SD 2 mbps
- VC1 SD to MPEG2 (TS&PS) SD 2 mbps

Successfully transcode a High Definition (HD) video to a Standard Definition video from and to the following formats:

- H264 720p to MPEG2 (TS&PS) SD 2 mbps
- H264 720p to VC1 SD 2 mbps

Successfully transcode a High Definition (HD) video to another HD video from and to the following formats:

- VC1 720p to H264 720p 7 mbps

Optional

Successfully transcode a High Definition (HD) video to another HD video from and to the following formats:

- H264 1080p to H264 720p 7 mbps
- VC1 1080p to H264 720p 7 mbps
- H264 720p to MPEG2 (TS&PS) 720p 7 mbps
- H264 720p to VC1 720p 7 mbps

Successfully transcode a High Definition (HD) video to another HD video from and to the following formats:

- H264 1080p to MPEG2 (TS&PS) 720p 7 mbps

Design Notes:

- MPEG-2 support is required on x86 and x64 architectures and operating systems only.
- See the Windows Driver Kit, Streaming Devices (Video and Audio), Hardware MFT Device Class and Stream Class Minidrivers.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.MobileBroadBand

These are requirements for Mobile Broadband devices integrated in the systems.

Related Requirements	• System.Client.MobileBroadBand.ClassDriver • System.Client.MobileBroadBand.ConcurrentRadioUsage • System.Client.MobileBroadBand.MobileBroadBand

System.Client.MobileBroadBand.ClassDriver

USB interface based GSM and CDMA class of Mobile Broadband device firmware must comply with USB-IF's Mobile Broadband Interface Model Specification.

Target Feature	System.Client.MobileBroadBand
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

USB interface based GSM and CDMA class of Mobile Broadband device firmware implementation must comply with the USB-IF's Mobile Broadband Interface Model (MBIM) Specification. No additional IHV drivers are needed for the functionality of the device and the device must work with Microsoft's Mobile Broadband(MB) class driver implementation. Note that Microsoft generic class driver doesn't support non-USB interface devices. Non-USB based devices require device manufacturer's device driver compliant with MB driver model specification.

Additional Details:
Mobile Broadband Interface Model Specification: http://www.usb.org/developers/devclass_docs/MBIM10.zip
Mobile Broadband Driver Model Specification: http://msdn.microsoft.com/en-us/library/windows/hardware/ff560543(v=VS.85).aspx

Exceptions:
- Device models that are announced as End of life (EOL) as of December, 2011.
- Device models that are no longer in production line.

Note that above exceptions are applicable only if:
- devices are used in Windows 8 Client x86 and Windows 8 Client x64.
- devices are pre-certified for multiple operators (at least 20).

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.MobileBroadBand.ConcurrentRadioUsage

System Builders must ensure that the RF performance is optimized for Mobile Broadband, Wi-Fi and Bluetooth radios running at the same time

Target Feature	System.Client.MobileBroadBand
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

System Builders must ensure that the RF performance is optimized for Mobile Broadband, Wi-Fi and Bluetooth radios running at the same time. Systems that enable internet connection sharing (tethering), multi-homing, and network switching all require multiple radios to be active simultaneously. Systems should ensure high throughput, high reliability, optimal power efficiency and minimum RF interference under these conditions regardless of the system form factor.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.MobileBroadBand.MobileBroadBand

Systems that include Broadband support meet Windows requirements

Target Feature	System.Client.MobileBroadBand
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Firmware Requirements

USB based devices for GSM and CDMA technologies (3GPP/3GPP2 standards based) need to be firmware compliant with the Mobile Broadband Interface Model specification. These devices need to be certified by the USB Forum for compliance (when it becomes available for MB devices).

In addition to the above, firmware needs to support the features listed below as specified by NDIS.

Firmware Feature	Requirement
No Pause on Suspend	Required
USB Selective Suspend	Required – If USB based
Radio Management	Required
Wake on Mobile Broadband	Required
Fast Dormancy	Required

No additional Connection Manager software is required for the operation of mobile broadband devices.

Value-add Mobile Broadband Connection Managers, if implemented, need to implement the Mobile Broadband API (http://msdn.microsoft.com/en-us/library/dd323271(VS.85).aspx).

Microsoft strongly recommends USB-based bus interfaces such as analog USB, HSIC (where applicable) and SSIC (when available). Mobile Broadband stack in Windows 8 is designed to support only USB protocol based bus interfaces. The following table summarizes the required mobile broadband features.

Attribute	Requirement
Bus	USB-HSCI (preferred) or USB

Systems must also comply with Mobile Broadband requirements, with:

- Devices MUST support 16 bitmap wake patterns of 128 bytes each.
- Devices MUST wake the system on register state change.
- Devices MUST wake the system on media connect.
- Devices MUST wake the system on media disconnect.
- GSM and CDMA class of Devices MUST wake the system on receiving an incoming SMS message.
- Devices that support USSD MUST wake the system on receiving USSD message.
- Devices MUST support wake packet indication. NIC should cache the packet causing the wake on hardware and pass it up when the OS is ready for receives.

Mobile Broadband class of devices must support Wake on Mobile Broadband. It should wake the system on above mentioned events. Note that wake on USSD is mandatory only if the device reports that it supports USSD. Else it is optional. See the following MSDN documentation for more information on the SMS and register state wake events.

- NDIS_STATUS_WWAN_REGISTER_STATE
- NDIS_STATUS_WWAN_SMS_RECEIVE

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.NearFieldProximity

Near Field Proximity is a set of short range wireless technologies to enable communication between a personal computer and a device.

Related Requirements	• System.Client.NearFieldProximity.ImplementingProximity • System.Client.NearFieldProximity.RangeOfActuation • System.Client.NearFieldProximity.TouchMark

System.Client.NearFieldProximity.ImplementingProximity

How/when to implement NFP technology

Target Feature	System.Client.NearFieldProximity
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Any system that incorporates NFC technology (namely any technology that utilizes one or more of the air interface specifications incorporated by the NFC Forum by reference as approved specifications) must provide an NFP provider for that NFC implementation that meets the Device.BusController.NearFieldProximity.ProviderImplementation and Device.BusController.NearFieldProximity.NFCCertification requirements.

Any system that incorporates any NFP technology that implements the device driver interface specified by the Device.BusController.NearFieldProximity.ProviderImplementation requirement must also provide an additional NFP provider that implements both the Device.BusController.NearFieldProximity.ProviderImplementation and Device.BusController.NearFieldProximity.NFCCertification requirements.

Any system that does not incorporate NFC technology and does not incorporate NFP technology that implements the device driver interface specified by the Device.BusController.NearFieldProximity.ProviderImplementation requirement need not meet NFP certification requirements.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.NearFieldProximity.RangeOfActuation

For devices using active radios, proximity technology meets range of actuation

Target Feature	System.Client.NearFieldProximity
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

An NFP provider must support an effective operating volume to enable users to successfully use NFP technology with Windows in 95 times out of 100 attempts for all tap and do scenarios. Refer to the most current version of the 'Windows 8 Near Field Proximity Implementation Specification' document for detailed placement guidance, as

well as acceptable, minimum, and maximum values for the required effective operating volume. The spec can be found at:

http://go.microsoft.com/fwlink/?LinkId=237135

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.NearFieldProximity.TouchMark

If the system has a proximity technology then there must be a mark to indicate where to tap devices together.

Target Feature	System.Client.NearFieldProximity
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

To help users locate and use the proximity technology, the use of a visual mark is required by, and is only permitted for, NFC NFP providers (those NFP providers that implement air interface specifications incorporated by the NFC Forum by reference as approved specifications).

Implementation can include but not limited to etching, inking and removable stickers. The mark that is used is at the discretion of the system builder.

Additional Information

Business Justification	The visual touch mark helps end users locate the antenna for NFC radio as the placement of the radio can vary based off of the form factor.
Enforcement Date	Jan. 03, 2012

System.Client.PCContainer

Starting with Windows 7, Windows is moving towards a device centric presentation of computers and devices. Elements of the Windows user interface (UI), such as the Devices and Printers folder, will show the computer and all devices that are connected to the computer. The requirements in this section detail what is required to have the PC appear as a single object in the Windows UI.

Related Requirements	• System.Client.PCContainer.PCAppearsAsSingleObject

System.Client.PCContainer.PCAppearsAsSingleObject

Computers must appear as a single object in the Devices and Printers folder

Target Feature	System.Client.PCContainer
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64

Description

Computers must appear as a single object in the Devices and Printers folder. Windows 7 and newer has a platform layer which groups all functionality exposed by the computer into a single object. This object is referred to as the computer device container. The computer device container must contain all of the device functions that are located physically inside the computer chassis. This includes, but is not limited to, keyboards, touch-pads; media control/media transport keys, wireless radios, storage devices, and audio devices. The computer device container is used throughout the Windows platform and is visibly exposed to the user in the Devices and Printers user interface. This requirement ensures a consistent and high quality user experience by enforcing the "one object per physical device" rule in the Devices and Printers folder.

The computer must appear as a single device container in the Devices and Printers folder for the following reason:

- Devices and Printers will be unable to provide a logical and understandable representation of the computer to the user. Accurate information as to which devices are physically integrated with the computer must be supplied to support this and dependent Windows features.

Design Notes:

Windows is moving towards a device centric presentation of computers and devices. The Devices and Printers folder will show the computer and all devices that are connected to the computer. In Devices and Printers the computer is represented by a single icon. All of the functionality exposed by the computer will be available through this single icon object, providing one location for users to discover devices integrated with the computer and execute specific actions on those integrated devices. To enable this experience, the computer must be able to detect and group all computer integrated devices. (I.e. all devices physically inside the PC.) This requires that computer integrated devices properly identify themselves as integrated components. This can be achieved by indicating that the device is not removable from computer, properly configuring ACPI for the port to which the device is attached, or creating a registry DeviceOverride entry for the device. (Note: Each bus type has different mechanisms for identifying the removable relationship for devices attached to that bus. Refer to the "Multifunction Device Support and Device Container Groupings in Windows 7" whitepaper for details.)

To group the functionality exposed by the computer into a single device container, Windows uses information available in the device hardware, bus driver, and system UEFI or BIOS and Windows registry. The bus type to which a given device is attached determines the heuristic Windows applies to group that device. The whitepaper titled "Multifunction Device Support and Device Container Groupings in Windows 7," which can be found at http://www.microsoft.com/whdc/Device/DeviceExperience/ContainerIDs.mspx, explains the heuristic for many bus types, including:

- Universal Serial Bus (USB)
- Bluetooth

- IP connected devices using Plug and Play Extensions (PnP-X)
- 1394
- eSATA
- PCI Express (PCIe)

The Single Computer Display Object test (ComputerSingleDDOTest.exe) must be executed on the system to check if this requirement has been met. The tool is available in Windows Logo Kit. The computer configuration for this test is different for laptops and desktop computers:

- Laptops: No external devices (other than a monitor) can be attached to the laptop when the ComputerSingleDDOTest.exe is run. If external devices of any type are attached the test will fail automatically.
- Desktops (computers which require an external keyboard, mouse and monitor to be attached): Only a keyboard, mouse and monitor can be attached to the desktop when the ComputerSingleDDOTest.exe is run. If any devices other than these are attached the test will fail automatically.

The ComputerSingleDDOTest.exe will identify those devices which Windows was unable to group with the computer device container. Determine the bus to which the indicated device is attached and follow the details in the whitepaper to determine why the device was not correctly grouped with the computer.
The design changes required to group an internal device with the computer device container vary depending on the bus to which the device is attached. Refer to the "Multifunction Device Support and Device Container Groupings in Windows 7" whitepaper for details.

Testing Notes:

This requirement can be tested by using the Single Computer Display Object Test (ComputerSingleDDOTest.exe) in the WLK. The computer configuration for this test is different for laptops and desktop computers:

- Laptops: No external devices can be attached to the laptop when the ComputerSingleDDOTest.exe is run. If external devices of any type are attached the test will fail automatically.
- Desktops (computers which require an external keyboard, mouse and monitor to be attached): Only a keyboard, mouse and monitor can be attached to the desktop when the ComputerSingleDDOTest.exe is run. If any devices other than these are attached the test will fail automatically.

The following are examples of the output from the Single Computer Display Object Test, showing both an unsuccessful and successful test pass. Both were executed on the same laptop computer. There were not any devices attached to the laptop at the time the test was executed.

Unsuccessful test result:

In the failure case, the laptop has two USB devices which are detected as separate device containers from the computer device container. These two devices are actually internal to the computer and connected to an internal USB hub.
D:\Tools\ComputerSingleDDOTest>ComputerSingleDDOTest.exe -laptop
Start: SystemFund-0200: Computers must be represented by one icon in Device Center., TUID=0E7AEA02-2712-422F-A9CD-FFBE470FD391
Found Device "USB Input Device". - This device should be part of the computer.

Found Device "USB Composite Device". - This device should be part of the computer.
Error: 0x0, Error 0x00000000
FAIL: Devices were found that are part of the computer but were reported
as removable from the computer.
File=__FILE__ Line=468
End: Fail, SystemFund-0200: Computers must be represented by one icon in Device
Center., TUID=0E7AEA02-2712-422F-A9CD-FFBE470FD391, Repro= D:\Tools\ComputerSingleDDOTest
\ComputerSingleDDOTest.exe
Summary: Total=1, Passed=0, Failed=1, Blocked=0, Warned=0, Skipped=0

Successful test result:

This is the same laptop as the unsuccessful test result above. The two USB devices are now grouped into the computer device container. This was done by creating a DeviceOverride registry entry for each device. Various options are available to achieve the correct grouping, depending on the bus type to which the device is attached. See "Multifunction Device Support and Device Container Groupings in Windows 7" for details.
D:\Tools\ComputerSingleDDOTest>ComputerSingleDDOTest.exe -laptop
Start: SystemFund-0200: Computers must be represented by one icon in Device Center., TUID=0E7AEA02-2712-422F-A9CD-FFBE470FD391
PASS: Devices were correctly reported as part of the computer.
End: Pass, SystemFund-0200: Computers must be represented by one icon in Device
Center., TUID=0E7AEA02-2712-422F-A9CD-FFBE470FD391, Repro=D:\Tools\ComputerSingl
eDDOTest\ComputerSingleDDOTest.exe
Summary: Total=1, Passed=1, Failed=0, Blocked=0, Warned=0, Skipped=0

Additional Information

Business Justification	The devices and printers folder will be unable to provide a logical and understandable representation of the PC without accurate information as to which devices are physically integrated with the PC.
Enforcement Date	Mar. 01, 2012

System.Client.PrecisionTouchpad

Precision Touchpad requirements applicable for Windows 8.1. Windows Precision Touchpads are a new class of input device deeply integrated in to the Windows platform to provide a consistent, reliable and high-performing user experience. A Windows precision touchpad is an integrated device that is internally connected as part of a clamshell/convertible system or as part of an attachment that provides both keyboard and touchpad functionality.

Related Requirements	• System.Client.PrecisionTouchpad.PrecisionTouchpad • System.Client.PrecisionTouchpad.RequiredForARM

System.Client.PrecisionTouchpad.PrecisionTouchpad

Precision Touchpad

Target Feature	System.Client.PrecisionTouchpad
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The following Precision Touchpad device level requirements must be met and verified upon integration into a system. Please refer to the following **Device.Input.PrecisionTouchpad** requirements for full requirement details:

Device.Input.PrecisionTouchpad.Buffering

Device.Input.PrecisionTouchpad.BusType

Device.Input.PrecisionTouchpad.FieldFirmwareUpdateable

Device.Input.PrecisionTouchpad.ThirdPartyDrivers

Device.Input.PrecisionTouchpad.WakeFunctionality

Device.Input.PrecisionTouchpad.WakeSource

Device.Input.PrecisionTouchpad.Hardware.Bezel

Device.Input.PrecisionTouchpad.Hardware.ClickpadPress

Device.Input.PrecisionTouchpad.Hardware.PressurePadPress

Device.Input.PrecisionTouchpad.I2C.ActivePowerConsumption

Device.Input.PrecisionTouchpad.I2C.ActiveToIdleTimeout

Device.Input.PrecisionTouchpad.I2C.BusSpeed

Device.Input.PrecisionTouchpad.I2C.ColdBootLatency

Device.Input.PrecisionTouchpad.I2C.ConnectedStandbyPowerConsumption

Device.Input.PrecisionTouchpad.I2C.IdlePowerConsumption

Device.Input.PrecisionTouchpad.Performance.ActiveTouchdownLatency

Device.Input.PrecisionTouchpad.Performance.IdleTouchdownLatency

Device.Input.PrecisionTouchpad.Performance.MinSeparation

Device.Input.PrecisionTouchpad.Performance.PanLatency

Device.Input.PrecisionTouchpad.Performance.ReportRate

Device.Input.PrecisionTouchpad.Precision.DiagonalInputSeparation

Device.Input.PrecisionTouchpad.Precision.EdgeDetection

Device.Input.PrecisionTouchpad.Precision.HVInputSeparation

Device.Input.PrecisionTouchpad.Precision.InputResolution

Device.Input.PrecisionTouchpad.Precision.Linearity

Device.Input.PrecisionTouchpad.Precision.MotionJitter

Device.Input.PrecisionTouchpad.Precision.Position

Device.Input.PrecisionTouchpad.Precision.StationaryJitter

Device.Input.PrecisionTouchpad.Reliability.FalseContacts

Device.Input.PrecisionTouchpad.Reliability.PowerStates

Device.Input.PrecisionTouchpad.USB.ActivePowerConsumption

Device.Input.PrecisionTouchpad.USB.BusSpeed

Device.Input.PrecisionTouchpad.USB.ColdBootLatency

Device.Input.PrecisionTouchpad.USB.IdlePowerConsumption

Device.Input.PrecisionTouchpad.USB.SelectiveSuspend

Device.Input.PrecisionTouchpad.USB.SleepPowerConsumption

Additional Information

Business Justification	To ensure Precision Touchpads which are integrated into a system maintain the expected user experience, bus, mechanical, performance, and reliability qualities.
Enforcement Date	Jun. 26, 2013

System.Client.PrecisionTouchpad.RequiredForARM

If a system with an ARM processor has a touchpad, it must be a Precision Touchpad

Target Feature	System.Client.PrecisionTouchpad
Applies to	• Windows 8.1 Client ARM (Windows RT 8.1)

Description

The precision touch pad on an ARM system must meet device requirements under the feature and sub features Device.Input.PrecisionTouchPad.

Additional Information

Business Justification	Windows precision touchpads provide a consistent user experience.
Enforcement Date	Jun. 26, 2013

System.Client.RadioManagement

This feature contains requirements for buttons that control the management of any radios in a laptop or Tablet/convertible PC. It also contains requirements for GPS radios, Near Field Proximity radios, and Bluetooth radios that do not use the Windows native Bluetooth stack.

Related Requirements	• System.Client.RadioManagement.HardwareButton • System.Client.RadioManagement.RadioMaintainsState • System.Client.RadioManagement.RadioManagementAPIHID • System.Client.RadioManagement.RadioManagerCOMObject

System.Client.RadioManagement.HardwareButton

If a PC has a physical (hardware) button switch on a PC that turns wireless radios on and off, it must be software controllable and interact appropriately with the Radio Management UI

Target Feature	System.Client.RadioManagement
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

There does not need to be a hardware button for wireless radios on Windows 8 laptops or tablet/convertible PCs.

A wireless hardware button is one of the following:

- Toggle Button (Laptops and Tablets)
- Toggle Button with LED (Non-Connected standby supported laptops and tablets)
- A-B slider switch (Laptops and Tablets)
- A-B slider switch with LED (Non-Connected standby supported laptops and tablets)

When there is a hardware button for wireless radios there **must not be more than one, and it must control all the radios present in the computer.**

An LED to indicate the state of the switch is optional. Please note that an LED indicating wireless status is not allowed on systems that support connected standby. If an LED is present along with the button, it must behave as defined below.

- There must only be one LED to indicate wireless status (i.e. there must be not one LED for Bluetooth, one for Wi-Fi, etc.).
- If the global wireless state is ON, the LED must be lit.
- When the global wireless state is OFF, the LED must not be lit.
- When the button is pressed or switch is flipped, it must send a HID message that can be consumed by the Radio Management API
- When the Radio Management API sends a HID message, the button or switch must receive the message and change the state of the LED accordingly.

Additional Information

Business Justification	The radio management feature goal is to create a consistent and predictable Windows experience for controlling all wireless capabilities on the PC that enables global as well as granular control of the radios. The UI is tightly integrated with the hardware switch so that the radio state is always accurate. The wireless hardware button must meet this requirement in order to meet these goals.
Enforcement Date	Mar. 01, 2012

System.Client.RadioManagement.RadioMaintainsState

Radio maintains on/off state across sleep and reboot power cycles

Target Feature	System.Client.RadioManagement
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The state of the wireless radio must persist across sleep, reboot, user log off, user switching and hibernate.

Additional Information

Business Justification	The radio management feature goal is to create a consistent and predictable Windows experience for controlling all wireless capabilities on the PC that enables global as well as granular control of the radios. The UI is tightly integrated with the hardware switch so that the radio state is always accurate. The wireless hardware button must meet this requirement in order to meet these goals.
Enforcement Date	Mar. 01, 2012

System.Client.RadioManagement.RadioManagementAPIHID

Wireless hardware button must communicate the change of state to the Radio Management API using HID

Target Feature	System.Client.RadioManagement
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

When the state of wireless radio switch changes, whether it is a slider A-B switch (with or without LED) or toggle button (with or without LED), this HID-compliant hardware switch/button must expose the HID collections to be consumed by the radio management API.

Toggle button must not change the state of the device radio directly.

A-B switch can be wired directly to the radios and change their state **as long as** it communicates the change of state to the Radio Management API using the HID driver and it changes the state in **all** radios present in the PC.

The HID usage IDs are:

Usage ID	Usage Name	Usage Type
0x0C	Wireless Radio Controls	CA
0xC6	Wireless Radio Button	OOC
0xC7	Wireless Radio LED	OOC
0xC8	Wireless Radio Slider Switch	OOC

The collections are:

Button without LED (stateless button) – For laptops, tablets and convertibles

```
USAGE_PAGE (Generic Desktop)          05 01
USAGE (Wireless Radio Controls)       09 0C
COLLECTION (Application)              A1 01
  LOGICAL_MINIMUM (0)                 15 00
  LOGICAL_MAXIMUM (1)                 25 01
  USAGE (Wireless Radio Button)       09 C6
  REPORT_COUNT (1)                    95 01
  REPORT_SIZE (1)                     75 01
  INPUT (Data,Var,Rel)                81 06
  REPORT_SIZE (7)                     75 07
  INPUT (Cnst,Var,Abs)                81 03
END_COLLECTION                        C0
```

Button with LED – For laptops, tablets and convertibles that do NOT support connected standby

```
USAGE_PAGE (Generic Desktop)          05 01
USAGE (Wireless Radio Controls)       09 0C
COLLECTION (Application)              A1 01
  LOGICAL_MINIMUM (0)                 15 00
  LOGICAL_MAXIMUM (1)                 25 01
  USAGE (Wireless Radio Button)       09 C6
  REPORT_COUNT (1)                    95 01
  REPORT_SIZE (1)                     75 01
  INPUT (Data,Var,Rel)                81 06
  REPORT_SIZE (7)                     75 07
  INPUT (Cnst,Var,Abs)                81 03
  USAGE (Wireless Radio LED)          09 C7
  REPORT_SIZE (1)                     75 01
  OUTPUT (Data,Var,Rel)               91 02
  REPORT_SIZE (7)                     75 07
  OUTPUT (Cnst,Var,Abs)               91 03
END_COLLECTION                        C0
```

Slider Switch (without LED) - For laptops, tablets and convertibles

```
USAGE_PAGE (Generic Desktop)          05 01
USAGE (Wireless Radio Controls)       09 0C
```

```
COLLECTION (Application)                A1 01
  LOGICAL_MINIMUM (0)                   15 00
  LOGICAL_MAXIMUM (1)                   25 01
  USAGE (Wireless Radio Slider Switch)    09 C8
  REPORT_COUNT (1)                      95 01
  REPORT_SIZE (1)                       75 01
  INPUT (Data,Var,Abs)                  81 02
  REPORT_SIZE (7)                       75 07
  INPUT (Cnst,Var,Abs)                  81 03
END_COLLECTION                          C0
```

Slider Switch with LED- Laptops, tablets and convertibles that do NOT support connected standby

```
USAGE_PAGE (Generic Desktop)            05 01
USAGE (Wireless Radio Controls)         09 0C
COLLECTION (Application)                A1 01
  LOGICAL_MINIMUM (0)                   15 00
  LOGICAL_MAXIMUM (1)                   25 01
  USAGE (Wireless Radio Slider Switch)    09 C8
  REPORT_COUNT (1)                      95 01
  REPORT_SIZE (1)                       75 01
  INPUT (Data,Var,Abs)                  81 02
  REPORT_SIZE (7)                       75 07
  INPUT (Cnst,Var,Abs)                  81 03
  USAGE (Wireless Radio LED)              09 C7
  REPORT_SIZE (1)                       75 01
  OUTPUT (Data,Var,Rel)                   91 02
  REPORT_SIZE (7)                       75 07
  OUTPUT (Cnst,Var,Abs)                   91 03
END_COLLECTION                          C0
```

LED Only (No button or slider) - Laptops, tablets and convertibles that do NOT support connected standby

```
USAGE_PAGE (Generic Desktop)              05 01
USAGE (Wireless Radio Controls)           09 0C
COLLECTION (Application)                  A1 01
  LOGICAL_MINIMUM (0)                     15 00
  LOGICAL_MAXIMUM (1)                     25 01
  USAGE (Wireless Radio LED)                09 C7
  REPORT_COUNT (1)                        95 01
  REPORT_SIZE (1)                         75 01
  OUTPUT (Data,Var,Rel)                     91 02
  REPORT_SIZE (7)                         75 07
  OUTPUT (Cnst,Var,Abs)                     91 03
END_COLLECTION                            C0
```

Wireless radio LED must have a HID-compliant driver to reflect the state of the airplane mode switch located in the user interface. Wireless radio LED only uses HID for output (no input since there is no button).

When the Radio Management API sends a HID message because the global wireless state (airplane mode) has changed, the switch must consume this message and toggle the state.

For an A-B switch, the manufacturer's proprietary embedded controller must report the correct state of the switch

at all times by sending a HID message to the HID driver, including every time the PC is turned on back on. Reporting the state of the A-B switch when the computer is turned back on is especially important in the case that the switch changed states while the PC was in states S3/S4/S5.

Additional Information

Business Justification	The radio management feature goal is to create a consistent and predictable Windows experience for controlling all wireless capabilities on the PC that enables global as well as granular control of the radios. The UI is tightly integrated with the hardware switch so that the radio state is always accurate. The wireless hardware button must meet this requirement in order to meet these goals.
Enforcement Date	Mar. 01, 2012

System.Client.RadioManagement.RadioManagerCOMObject

There must be a radio manager COM object which registers and interacts with the Radio Management API

Target Feature	System.Client.RadioManagement
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Systems that include any of the following radio must create a radio manager COM Object for that radio:

- GPS
- Proximity-type radios such as NFC and TransferJet
- Bluetooth radio that doesn't use the Windows Native Bluetooth stack

Each radio that does not have an inbox radio manager must have its own IRadioInstance object and be enumerable from IMediaRadioManager::IEnumRadioInstance. There is inbox support for WLAN, mobile broadband and Bluetooth.

Verify IRadioInstance::GetRadioManagerSignature returns the media radio manager GUID.

RadioInstance::GetInstanceSignature must return device instance path.

The radio COM object manager must send and consume On and Off calls from the Radio Management API.

When the Airplane Mode switch is turned ON by the user, the Radio Management API calls the method OnSystemRadioStateChange(), with the sysRadioState parameter equal to SRS_RADIO_DISABLED, in which case the Media Radio Manager must record the current device radio state for later use and must set the radio state to DEVICE_SW_RADIO_OFF.

When the Airplane Mode switch is turned OFF by the user, the Radio Management API calls the method OnSystemRadioStateChange(), with the sysRadioState parameter equal to SRS_RADIO_ENABLED, in which case the device radio associated with the radio instance managed by this Media Radio Manager must transition to a previous device radio state (the last recorded state).

Radio COM object manager must report radio instance within 100 milliseconds. Please note that this does not mean that radio COM object manager has to be fully functional (ready to turning radio on/off); it just means that radio COM object manager needs to report the presence of a radio within 100 milliseconds.

Users do NOT need administrator privileges to change the radio state. Any standard user must be able to change the radio state.

Radio COM object manager must be able to change the state of the radio (on to off, or off to on) within 5 seconds.

Additional Information

Business Justification	The radio management feature goal is to create a consistent and predictable Windows experience for controlling all wireless capabilities on the PC that enables global as well as specific control of the radios. The UI is tightly integrated with the hardware switch so that the radio state is always accurate. The wireless hardware button must meet this requirement in order to meet these goals.
Enforcement Date	Jun. 26, 2013

System.Client.RadioManagement.ConnectedStandby

This feature contains requirements for buttons that control the management of any radios in a laptop or Tablet/convertible PC. The radios that this requirement applies to are GPS

Related Requirements	• System.Client.RadioManagement.ConnectedStandby.NoRadioStatusIndicatorLights

System.Client.RadioManagement.ConnectedStandby.NoRadioStatusIndicatorLights

Systems that support Connected Standby must not include a light indicating the status of the radios in the system

Target Feature	System.Client.RadioManagement.ConnectedStandby
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

In order to conserve energy, systems that support connected standby cannot include a status indicator indicating whether the radios are on.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.ScreenRotation

Screen rotation is the act of a user rotating a system from landscape to portrait and vice versa. The requirements in this section describe the behavior for the end user experience.

Related Requirements	• System.Client.ScreenRotation.SmoothRotation

System.Client.ScreenRotation.SmoothRotation

Systems with accelerometers perform screen rotation in 300 milliseconds and without any video glitches

Target Feature	System.Client.ScreenRotation
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

All Windows 8 systems with an accelerometer must have sufficient graphics performance to meet performance requirements for screen rotation.

- A WDDM driver is required to enumerate source modes for the integrated display. The WDDM driver must support rotated modes (0,90,180 and 270) for every mode that it enumerates for the integrated panel.
- The rotation is required to be supported even if the integrated panel is in a duplicate or extended topology with another display device. For duplicate mode, it is acceptable to rotate all targets connected to the rotated source. Per path rotation is allowed but not required.

Both the above mentioned requirements are optional for Stereo 3D capable resolutions.

The following performance metrics must be met:

Time to first frame	Glitching	Length of screen rotation
300 milliseconds	No glitching at 60 fps	300 milliseconds

Additional Information

Business Justification	Graphics performance is critical to deliver a good end-user experience on Windows 8. The performance of the HW subsystems and the graphics driver plays a big role in that.
Enforcement Date	Mar. 01, 2012

System.Client.Sensor

Related Requirements	• System.Client.Sensor.GNSSRFSensitivity • System.Client.Sensor.HumanProximitySensor • System.Client.Sensor.Integrated

System.Client.Sensor.GNSSRFSensitivity

GNSS RF Sensitivity

Target Feature	System.Client.Sensor
Applies to	• Windows 8.1 Client ARM (Windows RT 8.1)

Description

GPS device and antenna shall have Over the Air (OTA) acquisition sensitivity as -142 dB or better. For OTA tracking sensitivity, it should be better than -145 dB.

Interference with other components in the system shall not cause degradation from these sensitivity goals. Display, Camera, other radios e.g. NFC, Bluetooth, WiFi, Mobile Broadband are some of the potential components which can cause interference. Active usage of such components shall not degrade GPS RF sensitivity.

Human hands holding the device one of the common positions shall not degrade GPS RF Sensitivity. Device shall be able to maintain OTA acquisition sensitivity at -142 dBm and OTA tracking sensitivity of -145 dBm when the system is held in common positions.

Additional Information

Business Justification	Poor RF performance of the device or poor antenna selection, placement and wiring of the antenna can prevent GPS devices from getting a fix even in good signal conditions. GPS signals being weak when they reach to the ground and them being very prone to interference make GPS RF sensitivity particularly important in order to have well performing GPS devices.
Enforcement Date	Jun. 26, 2013

System.Client.Sensor.HumanProximitySensor

System with human proximity sensor meets Windows requirements

Target Feature	System.Client.Sensor
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Sensor Category: SENSOR_CATEGORY_BIOMETRIC
Sensor Type: SENSOR_TYPE_HUMAN_PROXIMITY

All human proximity class sensors need to ensure that they accurately report the following Data Types to be seamlessly integrated with Windows (through the sensors platform).

Data type	Type	Meaning
SENSOR_DATA_TYPE_HUMAN_PROXIMITY	**VT_R4**	Distance between a human and the computer, in meters.

Note: Sensor Connection Type = SENSOR_CONNECTION_TYPE_PC_INTEGRATED for hardware that is built-in to the PC enclosure. Note that proximity sensors with connection type = SENSOR_CONNECTION_TYPE_PC_ATTACHED can also be used for power management features (if integrated into connected peripheral).
For detailed information regarding sensor driver development, please see the Sensors topic in the Device and Driver Technologies section of the Windows Driver Kit (WDK).

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.Sensor.Integrated

If the system contains an internal sensor that is integrated with the Windows Sensor Platform, it must report itself to the sensors platform as an integrated and correctly oriented sensor

Target Feature	System.Client.Sensor
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

If a system contains the following sensors that are integrated with the Windows Sensor Platform, then each of these sensors should report itself to the operating system as an internally integrated sensor.

Sensor Connection Type = SENSOR_CONNECTION_TYPE_PC_INTEGRATED

Required sensor objects exposed via sensor platform (Applies only to Tablet and Convertible-Tablet form-factors):

Sensor	x86/x64	ARM
3D Accelerometer (SENSOR_TYPE_ACCELEROMETER_3D)	Optional	Optional
3D Gyrometer (SENSOR_TYPE_GYROMETER_3D)	Optional	Optional
3D Inclinometer (SENSOR_TYPE_INCLINOMETER_3D)	Optional***	Optional***
3D Compass (SENSOR_TYPE_COMPASS_3D)	Optional***	Optional***
Device Orientation (SENSOR_TYPE_AGGREGATED_DEVICE_ORIENTATION)	Optional	Optional
Ambient Light Sensor (SENSOR_TYPE_AMBIENT_LIGHT)	Optional	Optional

Required **Sensor fusion**** for PC-Integrated sensors (Applies only to Tablet and Convertible-Tablet form-factors):

Sensor	x86/x64	ARM

3D Compass • **3D accelerometer used for tilt compensation (required)** • **3D Gyrometer used to enhance data rate and data integrity (required)**	Optional***	Optional***
3D Inclinometer • **3D accelerometer and 3D Compass used to determine yaw, pitch, roll (required)** • **3D Gyrometer used to enhance data rate and data integrity (required)**	Optional***	Optional***
Device Orientation • **3D accelerometer and 3D Compass used to formulate Rotation Matrix, Quaternion (required)** • **3D Gyrometer used to enhance data rate and data integrity (required)**	Optional	Optional

****Sensor fusion** is the process of using data from multiple sensors to enhance existing sensor data, or to synthesize new sensor data types from raw sensor data

***** 3D Compass and 3D Inclinometer** are derived exclusively from the Device Orientation sensor in the Windows Runtime. Either of these exposed to Win32 COM API will be overwritten at the WinRT layer.

In addition, each of these sensors should be correctly configured and calibrated with proper orientation (for example: mounted in proper direction) as per the guidance specified for the specific sensor category as outlined in related white papers and documents.

Sensors should not raise events when the system is stationary and the environment is not changing.

For detailed information regarding sensor driver development, see Sensor Driver Design Guide in the Device and Driver Technologies section of the Windows Driver Kit (WDK).

Additional Information

Business Justification	To ensure the internal sensor reports data in the standardized windows Data Types. If a sensor does not report these data fields, it will not be treated as an internal sensor and will not be exposed to applications.
Enforcement Date	Aug. 01, 2012

System.Client.Sensor.Base

Base feature for requirements that are base and apply to all sensor types

Related Requirements	• System.Client.Sensor.Base.ALSCalibrationTest • System.Client.Sensor.Base.DataEvents • System.Client.Sensor.Base.GNSSTestProperties • System.Client.Sensor.Base.PowerState • System.Client.Sensor.Base.SupportDataTypesAndProperties

System.Client.Sensor.Base.ALSCalibrationTest

ALS Calibration Test

Target Feature	System.Client.Sensor.Base
Applies to	• Windows 8.1 Client ARM (Windows RT 8.1)

Description

ALS should report lux values within 10% accuracy when a 100-200 lux light source is aimed directly into the ALS aperture. ALS should report lux values within 50% attenuation when the light source is aimed at an angle of 35 degrees from the ALS aperture

Additional Information

Business Justification	There have been multiple cases where system designs did not account for "shadow effects" which occur when the ALS aperture is either too deep or too small in diameter. This causes incorrect light level readings which in turn cause Windows to incorrectly dim the screen.
Enforcement Date	Jun. 26, 2013

System.Client.Sensor.Base.DataEvents

Data Events

Target Feature	System.Client.Sensor.Base
Applies to	• Windows 8.1 Client ARM (Windows RT 8.1)

Description

If a client sets the change sensitivity to 0, or when the change sensitivity is not applicable (e.g. for location), data events shall not fire more often than 80% of the CRI . The device also shall not miss more than 5% of the data reports , both for short (e.g. <= 1 sec) and long (e.g. > 1 minute) values of the current report interval.

Data events should not be fired if the current report interval and change sensitivity (when applicable) are not met.

Data events should not be fired at a rate less than the minimum report interval.

Additional Information

Business Justification	If data events are not fired when change sensitivity or current report interval values are satisfied there can be power and performance degradation. These tests make sure that sensors fire data when they are expected to.
Enforcement Date	Jun. 26, 2013

System.Client.Sensor.Base.GNSSTestProperties

Location GNSS Test Properties

Target Feature	System.Client.Sensor.Base
Applies to	• Windows 8.1 Client ARM (Windows RT 8.1)

Description

GPS devices shall support Assisted GPS (A-GPS). It is up to the device which A-GPS solution e.g. internet based, mobile operator based solution to implement.

While the GPS driver can utilize the other devices on the system for enhancements e.g. faster Time-To-Fix (TTF), GPS device shall continue to function when such devices are disabled, went into low power states or their radios are turned off.

In order to enable AGPS testing and cold starting the device when needed, GNSS Drivers must support SENSOR_PROPERTY_CLEAR_ASSISTANCE_DATA.

In order to allow turning on and turning off NMEA sentences in data reports, GNSS Driver must support SENSOR_PROPERTY_TURN_ON_OFF_NMEA. NMEA lines shall not be included in data reports by default.

```
//{e1e962f4-6e65-45f7-9c36-d487b7b1bd34}

DEFINE_GUID(SENSOR_PROPERTY_TEST_GUID,
0XE1E962F4, 0X6E65, 0X45F7, 0X9C, 0X36, 0XD4, 0X87, 0XB7, 0XB1, 0XBD, 0X34);

DEFINE_PROPERTYKEY(SENSOR_PROPERTY_CLEAR_ASSISTANCE_DATA,             0XE1E962F4,
0X6E65, 0X45F7, 0X9C, 0X36, 0XD4, 0X87, 0XB7, 0XB1, 0XBD, 0X34, 2); //[VT_UI4]

DEFINE_PROPERTYKEY(SENSOR_PROPERTY_TURN_ON_OFF_NMEA,             0XE1E962F4,
0X6E65, 0X45F7, 0X9C, 0X36, 0XD4, 0X87, 0XB7, 0XB1, 0XBD, 0X34, 3); //[VT_UI4]

#define GNSS_CLEAR_ALL_ASSISTANCE_DATA 0x00000001
```

SENSOR_PROPERTY_ CLEAR_ASSISTANCE_DATA (PID = 2)	**VT_UI4** **Write. The assistance data to be cleared.** **Setting a value of GNSS_CLEAR_ALL_ASSISTANCE_DATA signals the driver to clear all assistance data, including time, almanac, ephemeris and last position.** **WHCK tests can set this value to clear the assistance data before a cold start test, AGPS tests or independently before running simulator tests where time and location is simulated. If A-GPS capabilities e.g. SUPL, LTO is supported, driver can try to utilize them after this operation by using the network connection. However, it should be in a state where no assistance data is saved in the device or on the system. Any assistance data elements shall be downloaded again.**
SENSOR_PROPERTY_TURN_ON_OFF_NMEA (PID = 3)	**VT_UI4** Read/Write. If set to TRUE, NMEA sentence shall be included in data reports. If set to False, NMEA sentence shall not be included in data reports. WHCK tests can use this property to instruct the device to start sending NMEA data or stop including it in data reports.

Additional Information

Business Justification	In order to test Time-To-Fix performance characteristics of a GPS device e.g. whether A-GPS is supported or Time-To-First-Fix after a “cold start”, it is needed to instruct the driver to clear the assistance data. Since there is no user scenario for apps to clear the assistance data, which will increase the time to fix, this is a test only property and is not exposed via API. Adding NMEA sentence with all data reports notably increases the data being passed

	to upper layers. SENSOR_PROPERTY_TURN_ON_OFF_NMEA allows the test app to ask for inclusion of this data only when needed.
Enforcement Date	Jun. 26, 2013

System.Client.Sensor.Base.PowerState

Power State

Target Feature	System.Client.Sensor.Base
Applies to	• Windows 8.1 Client ARM (Windows RT 8.1)

Description

All sensors shall enter a low power state (D2 or D3) when all clients are disconnected and shall power up when a client is connected.

GPS drivers shall support Idle Detection. The Location Sensor driver for a GPS device must transition the device to the Sleep power mode whenever possible. If an app requests a report interval longer than 60 seconds, the Location Sensor driver should transition the GPS device to D3 until the next fix is requested. The Location Sensor driver must transition the GPS device to D0 with sufficient time to triangulate a fix and provide the app with location data.

GPS devices shall implement the power and sensor states described in following table:

ASIC state on = position computation in progress

Inputs				**Outputs**		
Client exists	Radio state	Long CRI	Position reported	ASIC state	Sensor state	Power state
No	Any	Any	Any	Off	-	D3
Yes	On	No	No	On	Initializing	D0
Yes	On	No	Yes	On	Ready	D0
Yes	Off	Any	Any	Off	Not available	D3
Yes	On	Yes	Any	Off	Ready	D3
Yes	On	Yes	Any	On	Ready	D0

Additional Information

Business Justification	If a sensor cannot enter a low power when all clients are disconnected, Windows will not be able to achieve power savings. If a sensor cannot power up when a client is connected, the machine cannot function properly.
Enforcement Date	Jun. 26, 2013

System.Client.Sensor.Base.SupportDataTypesAndProperties

Sensor and Location Platform devices support the set of data types and properties as defined in this requirement.

Target Feature	System.Client.Sensor.Base
Applies to	• Windows 8.1 Client ARM (Windows RT 8.1)

Description

All sensor devices that implement the Sensor Device Driver Interface must meet the following requirements. See any additional requirements that are specific to the sensor type being tested. For detailed information regarding sensor driver development, please see the Sensors topic in the Device and Driver Technologies section of the Windows Driver Kit (WDK).

Required Properties

- Sensor devices should show accurate data. The data being provided must follow the guidelines described in MSDN for each property and device type.
- These properties are queried using Device Driver Interfaces ISensorDriver::OnGetSupportedSensorObjects, ISensorDriver::OnGetSupportedProperties() and ISensorDriver::OnGetProperties(). Explicit type matching is required for data types and properties. The types must be the same as in the charts below.

Property	Data type	Static	Details
WPD_FUNCTIONAL_OBJECT_CATEGORY	VT_CLSID	Static	
SENSOR_PROPERTY_TYPE	VT_CLSID	Static	
SENSOR_PROPERTY_STATE	VT_UI4	Not static	
SENSOR_PROPERTY_PERSISTENT_UNIQUE_ID	VT_CLSID	Static	
SENSOR_PROPERTY_MANUFACTURER	VT_LPWSTR	Static	
SENSOR_PROPERTY_MODEL	VT_LPWSTR	Static	
SENSOR_PROPERTY_SERIAL_NUMBER	VT_LPWSTR	Static	
SENSOR_PROPERTY_FRIENDLY_NAME	VT_LPWSTR	Static	
SENSOR_PROPERTY_MIN_REPORT_INTERVAL	VT_UI4	Static	
SENSOR_PROPERTY_CONNECTION_TYPE	VT_UI4	Static	

Required Static Properties

- The following properties must not change value over time, so that the Sensor and Location Platform (and applications) can use them to select and manage sensors. These properties are queried using Device Driver Interfaces ISensorDriver::OnGetSupportedProperties() and ISensorDriver::OnGetProperties().
- Data types for these properties must match those in the following chart.
- The properties must be implemented following the guidelines described in MSDN.

Settable Properties

- Applications can use settable properties to configure the driver (such as optimize for power or other factors). Other settable properties can be exposed besides the following ones, but if this property is exposed, it must be settable.

- Setting SENSOR_PROPERTY_CURRENT_REPORT_INTERVAL < SENSOR_PROPERTY_MIN_REPORT_INTERVAL does not change the current report interval and setting SENSOR_PROPERTY_CHANGE_SENSITIVITY < 0 does not change the change sensitivity.

  The sensor shall be able to handle changes to the current report interval and change sensitivity for a single and multiple clients.

- The sensor should be able to respond to a client asking to be removed from calculation of the effective current report interval and change sensitivity. Clients can set CS to VT_NULL but there isn't an API to be removed. CRI/CS set by multiple clients needs to respect that most sensitive request.

  Sensors shall follow the current report interval and change sensitivity (when applicable) documentation found on MSDN:
  http://msdn.microsoft.com/en-us/library/windows/hardware/hh706201(v=vs.85).aspx

- GPS devices shall support property value for SENSOR_PROPERTY_MIN_REPORT_INTERVAL at one second or less and be capable of sending events at this interval.

Sensor drivers and firmware must support and adhere to the following Settable Properties: in the Sensor API.

Property	Data type	Details
SENSOR_PROPERTY_CURRENT_REPORT_INTERVAL	VT_UI4	Sets the minimum frequency (in milliseconds) that a client wants to receive data reports from the sensor. This property should be tracked on a per client basis.
SENSOR_PROPERTY_CHANGE_SENSITIVITY	VT_UNKNOWN	Sets the threshold of how much a data field must change before an event is fired.

The current report interval and change sensitivity retrieved must be within the tolerances listed below:

Single client (or multiple clients with only one client setting CRI):

Set Value	Expected value
0	Default CRI
0< value <(Min CRI)	Not changed, sensor reports error
(Min CRI) <= value (where value = (Min CRI)*N+T where T is 0<T<(Min CRI))	(Min CRI)*N <= value <= (Min CRI)*N + T where T is 0<T<(Min CRI)

Multiple clients:

Set Value	Expected value
0 (Default CRI)	No change if client set value greater than effective CRI. New effective CRI if based on CRI set by other clients. Effective CRI is smallest CRI of those set by clients but >= Min CRI
0< value <(Min CRI)	Not changed, sensor reports error
(Min CRI) <= value < effective CRI (where value = (Min CRI)*N+T where T is 0<T<(Min CRI))	(Min CRI)*N <= value <= (Min CRI)*N + T where T is 0<T<(Min CRI))
> effective CRI	Not changed

Data Fields

- Sensor devices are useful only when they report data. Each device must report at least one data field in addition to SENSOR_DATA_TYPE_TIMESTAMP (VT_FILETIME). Data fields are exposed to the Sensor API by using Device Driver Interface ISensorDriver::OnGetSupportedDataFields() and ISensorDriver::OnGetDataFields().
- GPS testing shall be performed in availability of GPS signal.
- GPS devices shall provide accurate latitude, longitude and altitude values within the specified error radius.
- GPS devices shall be able to report horizontal accuracy of 15 meters and vertical accuracy of 100 meters at 95% of the time under clear sky conditions. This applies to both static or mobile test scenarios.

  Clear sky conditions: GNSS satellites signals are received without obstruction from above or surrounding environment down to an elevation mask of 5 degrees above the horizon. All signal levels consistent with unobstructed signal levels at the ground and not to be lower than -131 dBm for 4 or more satellites.

- GPS devices shall be able to report speed in knots with +-20% accuracy.
- GPS devices shall be able to report true heading degrees with +-20% accuracy.

GPS Shall support the following data fields:

Data Field	Type
SENSOR_DATA_TYPE_LATITUDE_DEGREES	**VT_R8**
SENSOR_DATA_TYPE_LONGITUDE_DEGREES	**VT_R8**
SENSOR_DATA_TYPE_ERROR_RADIUS_METERS	**VT_R8**
SENSOR_DATA_TYPE_SATELLITES_USED_COUNT	**VT_I4**
SENSOR_DATA_TYPE_ALTITUDE_ELLIPSOID_METERS	**VT_R8**
SENSOR_DATA_TYPE_ALITITUDE_ELLIPSOID_ERROR_METERS	**VT_R8**
SENSOR_DATA_TYPE_SPEED_KNOTS	**VT_R8**
SENSOR_DATA_TYPE_TRUE_HEADING_DEGREES	**VT_R8**
SENSOR_DATA_TYPE_NMEA_SENTENCE	**VT_LPWSTR**
SENSOR_DATA_TYPE_SATELLITES_IN_VIEW_STN_RATIO	**VT_VECTOR | VT_UI1**

Location may support the following data fields. If they are supported, they must be implemented according to the guidelines in MSDN.

SENSOR_DATA_TYPE_ALTITUDE_SEALEVEL_METERS	VT_R8
SENSOR_DATA_TYPE_MAGNETIC_HEADING_DEGREES	**VT_R8**
SENSOR_DATA_TYPE_MAGNETIC_VARIATION	**VT_R8**
SENSOR_DATA_TYPE_FIX_QUALITY	**VT_I4**
SENSOR_DATA_TYPE_FIX_TYPE	**VT_I4**
SENSOR_DATA_TYPE_POSITION_DILUTION_OF_PRECISION	**VT_R8**
SENSOR_DATA_TYPE_HORIZONAL_DILUTION_OF_PRECISION	**VT_R8**
SENSOR_DATA_TYPE_VERTICAL_DILUTION_OF_PRECISION	**VT_R8**

SENSOR_DATA_TYPE_SATELLITES_USED_PRNS	**VT_VECTOR | VT_UI1**
SENSOR_DATA_TYPE_SATELLITES_IN_VIEW	**VT_I4**
SENSOR_DATA_TYPE_SATELLITES_IN_VIEW_PRNS	**VT_VECTOR | VT_UI1**
SENSOR_DATA_TYPE_SATELLITES_IN_VIEW_ELEVATION	**VT_VECTOR | VT_UI1**
SENSOR_DATA_TYPE_SATELLITES_IN_VIEW_AZIMUTH	**VT_VECTOR | VT_UI1**
SENSOR_DATA_TYPE_ADDRESS1	**VT_LPWSTR**
SENSOR_DATA_TYPE_ADDRESS2	**VT_LPWSTR**
SENSOR_DATA_TYPE_CITY	**VT_LPWSTR**
SENSOR_DATA_TYPE_STATE_PROVINCE	**VT_LPWSTR**
SENSOR_DATA_TYPE_POSTALCODE	**VT_LPWSTR**
SENSOR_DATA_TYPE_ALTITUDE_SEALEVEL_ERROR_METERS	**VT_R8**
SENSOR_DATA_TYPE_GPS_SELECTION_MODE	**VT_I4**
SENSOR_DATA_TYPE_GPS_OPERATION_MODE	**VT_I4**
SENSOR_DATA_TYPE_GPS_STATUS	**VT_I4**
SENSOR_DATA_TYPE_GEOIDAL_SEPARATION	**VT_R8**
SENSOR_DATA_TYPE_DGPS_DATA_AGE	**VT_R8**
SENSOR_DATA_TYPE_ALTITUDE_ANTENNA_SEALEVEL_METERS	**VT_R8**
SENSOR_DATA_TYPE_DIFFERENTIAL_REFERENCE_STATION_ID	**VT_I4**
SENSOR_DATA_TYPE_SATELLITES_IN_VIEW_ID	**VT_VECTOR | VT_UI1**

Missing Properties

- The device driver must correctly handle queries and sets for properties that a sensor device does not support. This enables the Sensor and Location Platform to correctly notify applications when sensor properties are missing.
- Drivers must return S_FALSE from Device Driver Interfaces ISensorDriver::OnGetProperties and ISensorDriver::OnSetProperties if one or more of the requested properties is not present on the device.
- For each missing property the PropVariant for the requested property must have a type of VT_ERROR and a value of HRESULT_FROM_WIN32(ERROR_NOT_FOUND). The driver can return valid values for properties it does have alongside not-valid values.

The follow requirements ensure the sensor drivers that receive a certification can report data reliably, obey the sensor driver state model and implement data timestamps correctly.

Reliability

- The driver must continue to operate after four hours of reporting data and "get data", "get property" calls.
- The driver must continue to operate when readable/writeable properties are set and read.
- The driver must provide data and properties after completion of sleep/resume or hibernation/resume cycle(s).
- The driver shall be able to handle concurrent PnP, radio (if GPS), power operations and multiple clients entering and exiting.

Timestamp

- The driver must report an accurate relative timestamp with each report.The timestamp shall be in Coordinated Universal Time (UTC) format. This timestamp must be greater than the initial prompt to provide a timestamp and must be less than or equal to the time the event is received.

Ready State Validation

Sensors shall not transition to the SENSOR_STATE_READY state until valid data is available.

Sensors must complete power-up initialization tasks and transition to the SENSOR_STATE_READY after being opened by a client or resuming from system standby within the following times:

Sensor Type	Maximum time allowed to enter ready state
Accelerometer	1 second
Gyro meter	3 seconds
Compass	3 seconds
Inclinometer	3 seconds
Orientation sensor	5 seconds
GPS	See following table

GPS Startup requirements under clear sky conditions

Startup type	Time elapsed from position request (in seconds)	Acceptable sensor states	Acceptable error radius
Cold start	0	SENSOR_STATE_INITIALIZING	n/a
	45	SENSOR_STATE_READY	Under 50 meters
Warm start (powered off with assistance data)	0	SENSOR_STATE_INITIALIZING	n/a
	10	SENSOR_STATE_READY	Under 300 meters
	20	SENSOR_STATE_READY	Under 50 meters
Hot start	0	SENSOR_STATE_INITIALIZING or SENSOR_STATE_READY depending if fix was maintained	Under 300 meters
	2	SENSOR_STATE_READY	Under 50 meters

Cold Start TTFF: Time Unknown, Current ephemeris unknown, position unknown

If GPS device supports D3 Cold, it should be able to gracefully go into and wake from D3 Cold. The overhead caused by waking form D3 Cold should not be more than 6 seconds for the First Fix after wake. This is in comparison from the wake from D3 Hot.

Hot Start TTFF: Time known, Almanac known, Ephemeris known, Position within 100km of last fix

GPS devices shall be able to acquire first fix under two seconds when 1) The radio is on, 2) Flight Mode is off, and 3) Under clear sky conditions.

GPS devices under clear sky conditions shall be able to report position from a cold start within 45 seconds.

A-GPS devices shall be able to report position within 10 seconds with 100-300 meter accuracy and within 20 seconds with 30 meter or less accuracy.

Location device shall be in initializing state after it is enabled and change to ready after acquiring the current position.

Location devices shall fire valid location data events with valid location data when in the ready state.

Location devices shall not fire data events when the device is not in either ready state or initializing state.Accuracy Requirements:

Sensor	Accuracy requirement
Accelerometer	+/- 0.1 G
Gyro	+/- 10 degrees per second square
Inclinometer	+/- 10 degrees
Compass	+/- 10 degrees
Orientation	Vector is +/- 15 degrees from true vector

Additional Information

Business Justification	To ensure the sensor drivers that receive a logo meet a high quality bar, it is necessary that the requirements include tests for the behavior of the device. The proposed additions ensure that the driver can report data reliably, obey the sensor driver state model and implement data timestamps correctly.
Enforcement Date	Jun. 26, 2013

System.Client.Sensor.Base.HID

Base feature for HCK requirements relating to HID, such as FW testing or type validation.

Related Requirements	• System.Client.Sensor.Base.HID.ReportDescriptor

System.Client.Sensor.Base.HID.ReportDescriptor

HID Report Descriptor

Target Feature	System.Client.Sensor.Base.HID
Applies to	• Windows 8.1 Client ARM (Windows RT 8.1)

Description

Any sensor device that uses the inbox SensorsHIDClassDriver.dll must be compliant with all recommendations in the Sensors HID Annotation document (http://msdn.microsoft.com/library/windows/hardware/br259128.aspx). In the course of executing the sensor, no ETW error events should be raised by the SensorsHIDClassDriver.dll.

Additional Information

Business Justification	The HID sensor is widely used and the underlying firmware must be checked for proper operation.
Enforcement Date	Jun. 26, 2013

System.Client.SpecializedPC

A Specialized PC (SPC) is a system that has been designed specifically for an enterprise vertical market or a niche use case scenario which has an explicit design need to bypass or remove features required by certification.

Related Requirements	• System.Client.SpecializedPC.UniqueScenario

System.Client.SpecializedPC.UniqueScenario

Speclalized PCs are designed for unique scenarios

Target Feature	System.Client.SpecializedPC
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

A Specialized PC (SPC) is a system that has been designed specifically for an enterprise vertical market or a niche use case scenario which has an explicit design need to bypass or remove features required by certification. An SPC may still be certified, provided there is adequate disclosure of the ways the system departs from the certification standards and it is clear what the end user impacts will be. A specialized PC is expected to meet all other certification requirements except those explicitly required to meet the use case and disclosed.

An SPC may be allowed to fail certain Windows Certification Requirements which conflict with the design needs of the specialized use. Details on the program are in the policy.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.SystemConfiguration

This feature and requirement defines the computers and devices that compose the system.

Related Requirements	• System.Client.SystemConfiguration.SysInfo • System.Client.SystemConfiguration.Windows7NeccessaryDevices • System.Client.SystemConfiguration.Windows8RequiredComponents

System.Client.SystemConfiguration.SysInfo

System information

Target Feature	System.Client.SystemConfiguration
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

A job in the Windows Hardware Certification Kit will collect information about the configuration of the machine. This will include information on:

- CPU Speed
- RAM
- Storage
- Hardware Resources
- Devices components
- Drivers
- Software

Additional Information

Enforcement Date	Mar. 01, 2012

System.Client.SystemConfiguration.Windows7NeccessaryDevices

Windows 7 systems includes necessary devices

Target Feature	System.Client.SystemConfiguration
Applies to	• Windows 7 Client x86, x64

Description

All buses, devices, and other components in a system must meet their respective Windows 7 logo program requirements and only use drivers that either are included with the Windows operating system installation media or are digitally signed by Microsoft for Windows 7. Driver files that are not signed for Windows 7 may not be included in the system image. Drivers submitted for the Windows 7 logo must be able to be digitally signed and must meet the Microsoft guidelines as defined in the Windows Driver Kit, "WHQL Digital Signature."

For a system to receive a Windows 7 logo it must comply with the Windows 7 logo requirements defined for a system and must include Windows 7 logo devices specified in Table B. These devices must comply with the Windows logo requirements (unless an exception is explicitly noted) as defined on LogoPoint.

TABLE B: System requirements
Systems must include the devices designated as R (required).

		Logo			
Group	Sub Group	Desktop	All-in-One	Mobile	Ultra Mobile
Graphics (3)		R	R	R	R
Display	Displays & Monitors (2)		R	R	R

Audio (1)	I	I	I	I
Storage (8)	R	R	R	R
Networking (6)	R	R	R	R

1) If audio is included in the system, it must meet all logo requirements.
2) Display on all-in-One and mobile systems must comply with requirements for Displays & Monitors.
3) Graphics solution in desktop, all-in-One and mobile systems must comply with requirements for graphics. **For Windows 7, desktop, mobile and all-in one systems must include a display adapter/chipset that complies with Direct3D version 10 and WDDM 1.1 beginning December 1, 2009**. Prior to December 1, 2009, WDDM v1.1 is Optional for a display adapter or chipset based on the Direct3D v9 hardware architecture and shipping in a Windows 7 system (desktop, mobile or all-in-one). Ultra Mobile PCs are required to ship with at least Direct3D version 9 and WDDM 1.0.
4) System must ship with a storage solution.
5) If system includes Audio it must be supported as a UAA compliant audio solution and be logo qualified.
6) Either WLAN or LAN device must be included in the system.
7) If any other devices are included in the system, they must be logo qualified for Windows 7
8) A storage drive is required for the OS. An optical disc drive is optional. Read only optical disc drives are still allowed to be included in logo'd systems after June 1, 2010.

Additional Information

Business Justification	Establishing a baseline configuration is important for the end user, as well as the development community. All systems must have graphics and storage. Networking is required in order to deploy updates to the system. All systems must also include a way to debug the system as document under System.Fundamentals.DebugPort.
Enforcement Date	Jun. 01, 2007

System.Client.SystemConfiguration.Windows8RequiredComponents

Windows 8 systems must include certain devices

Target Feature	System.Client.SystemConfiguration
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

For all other Windows 8 systems, the table below lists the minimum required components to be present in a system in order for it to be certified for Windows 8. All components must meet certification requirements and pass device certification testing for Windows 8.

Onscreen displays for the monitor are allowed. They can only appear after the user has taken action such as pressing a button. The inbox onscreen display cannot be duplicated.

Component		Required
Storage	Space	At least 10 GB of free space after completing OOBE on devices with greater than 16 GB of internal storage At least 5 GB of free space after completing OOBE on devices with 16 GB of

		internal storage or lower
	Storage type	Meet minimum Microsoft Windows Storage requirements
System firmware		UEFI as defined in System.Fundamentals.Firmware requirements
Networking	Ethernet or Wi-Fi	Must be either a certified Ethernet or Wi-Fi adapter
Graphics	GPU	Minimum of Direct3D 10 Feature Level 9_3 and see System.Fundamentals.Graphics.WDDM
	Video playback	See System.Client.VideoPlayback.WNGlitchfreeHDVideoPlayback
	Minimum resolution	See System.Fundamentals.Graphics.Display.MinimumResolutionandColorDepth

Additional Information

Business Justification	A Windows 8 system must include a network connection for browsing the Internet and for services the system. This can either be Wi-Fi radio hardware for Internet and local area network connectivity or an Ethernet adapter. The optimal physical screen size allows for 720-p HD video playback. The video output can be VGA or support any digital connection to connect to a monitor.
Enforcement Date	Mar. 01, 2012

System.Client.SystemImage

The requirements in this section describe the level 2 quality of HW + SW + OEM image

Related Requirements	• System.Client.SystemImage.PushButtonReset • System.Client.SystemImage.SystemRecoveryEnvironment

System.Client.SystemImage.PushButtonReset

ARM System includes push button reset to initial factory state

Target Feature	System.Client.SystemImage
Applies to	• Windows 8.1 Client ARM (Windows RT 8.1)

Description

An ARM system shall include a recovery image file that is compatible with the Push Button Reset feature in Windows 8. This recovery image file shall reside on a local recovery partition or on a separate bootable USB flash drive which is packaged with the system, or both.

If the recovery image file resides on a local recovery partition, the GPT partition shall be of type PARTITION_BASIC_DATA_GUID, and shall include the GPT_ATTRIBUTE_PLATFORM_REQUIRED and GPT_BASIC_DATA_ATTRIBUTE_NO_DRIVE_LETTER attributes.

If the recovery image file resides on a separate bootable USB flash drive, the drive shall be configured to load the Windows Recovery Environment when used as the boot device, and the recovery image file shall be stored in the \Sources folder under the root of the drive.

The recovery image file shall meet the following requirements:

1. The recovery image file shall be in the Window image file format (.wim)
2. The recovery image file shall be named "install.wim" (or in the case of a split image, "install.swm", "install2.swm", "install3.swm", and so on€¦)
3. The recovery image file shall contain the Windows image and applications preloaded on the system.
4. The Windows image in the recovery image file shall be configured to start Windows Welcome (instead of Audit mode) when it boots for the first time.
5. The recovery image file shall be captured after the Windows image has completed the Specialize configuration pass.
6. The Windows System Assessment Tests (WinSAT) results for the DWM test shall be prepopulated in the Windows image in the recovery image file.

Additional Information

Business Justification	Consumer electronics level of experience is expected. Users shall be able to refresh or reset their systems quickly and reliably.
Enforcement Date	Aug. 01, 2012

System.Client.SystemImage.SystemRecoveryEnvironment

System includes Windows Recovery Environment on a separate partition

Target Feature	System.Client.SystemImage
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

A system must include a separate partition with a bootable Windows Recovery Environment image file (winre.wim). The GPT partition should be of type PARTITION_MSFT_RECOVERY_GUID, includes the GPT_ATTRIBUTE_PLATFORM_REQUIRED and GPT_BASIC_DATA_ATTRIBUTE_NO_DRIVE_LETTER attributes, and contains at least 50 megabytes (MB) of free space after the Windows Recovery Environment image file has been copied to it.

Additional Information

Business Justification	Allows volume encryption features in Windows to be enabled on the operating system volume without affecting the Windows Recovery Environment.
Enforcement Date	Jun. 26, 2013

System.Client.SystemPartition

The requirements in this section describe the PC system partition configuration requirements.

Related Requirements	• System.Client.SystemPartition.DiskPartitioning • System.Client.SystemPartition.OEMPartition

System.Client.SystemPartition.DiskPartitioning

Systems that ship with a Windows operating system must meet partitioning requirements

Target Feature	System.Client.SystemPartition
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64

Description

A Windows 7 or Windows 8 system shipped in legacy BIOS mode and Master Boot Record (MBR) configured must ship with a type 0x7 or type 0x27, active system partition in addition to the operating system partition (configured as Boot, Page File, Crash Dump, etc.). This active system partition must be at least 100MB in size for Windows 7 systems. For Windows 8 systems, this active system partition must have at least 250MB of free space, above and beyond any space used by required files. For Windows 7 systems, we similarly recommend that the active partition have at least 250MB of free space for future upgrade to a new version of Windows. This additional system partition can be used to host Windows Recovery Environment (RE) and OEM tools (provided by the OEM), so long as the partition still meets the 250MB free space requirement.

Implementation of this partition allows support of current and future Windows features such as BitLocker, and simplifies configuration and deployments.
Tools and documentation to implement split-loader configuration can be found in **Windows OEM Preinstallation Kit/Automated Installation Kit (OPK/AIK).**

Additional Information

Business Justification	If the system is not partitioned correctly for BitLocker, the feature will not work. The user will not be able to turn the feature on as the feature has a dependency on having this second partition. This second partition is also required for the recovery story. When the owner of the platform loses cryptographic keys or experiences encryption or disk corruption, then the owner of the platform will lose their data.

Enforcement Date	Jun. 01, 2006

System.Client.SystemPartition.OEMPartition

Windows systems with recovery & OEM partitions must meet partitioning requirements

Target Feature	System.Client.SystemPartition
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

If a system includes a separate partition for recovery purposes or an OEM partition for any other purpose, this separate partition must be identified with the GPT_ATTRIBUTE_PLATFORM_REQUIRED attribute. This attribute is defined as part of the PARTITION_INFORMATION_GPT (http://msdn.microsoft.com/en-us/library/aa365449(VS.85).aspx) structure.

For example:

- If this separate partition includes a bootable Windows Recovery Environment image file, the GPT partition must be of type PARTITION_MSFT_RECOVERY_GUID and include the GPT_ATTRIBUTE_PLATFORM_REQUIRED attribute.
- If this separate partition includes a recovery image used by Push Button Reset, the GPT partition must be of type PARTITION_BASIC_DATA_GUID and include the GPT_ATTRIBUTE_PLATFORM_REQUIRED attribute.

Partitions which are identified with the GPT_ATTRIBUTE_PLATFORM_REQUIRED attribute must not be used for storing user data (such as through data backup, for example).

Additional Information

Business Justification	Allows volume encryption features in Windows 8 to be enabled on the operating system and data volumes without affecting the OEM and recovery partitions.
Enforcement Date	Mar. 01, 2012

System.Client.Tablet

The requirements apply to systems that are a tablet or tablet/convertible system.A tablet form factor is defined as a standalone device that can combine the PC, display that is 17-inches or smaller and rechargeable power source in a single chassis. A tablet does not include a permanently attached keyboard and pointing device but can be connected to a port replicator, keyboard and/or clamshell dock. A convertible form factor is defined as a standalone device that combines the PC, display that is 17-inches or smaller and rechargeable power source with a mechanically attached keyboard and pointing device in a single chassis. A convertible can be transformed into a tablet where the attached input devices are hidden or removed leaving the display as the only input mechanism.

Related Requirements	• System.Client.Tablet.ColdBootLatency • System.Client.Tablet.RequiredHardwareButtons

System.Client.Tablet.ColdBootLatency

Time for I2C touch controller to respond

Target Feature	System.Client.Tablet
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The time from when power is applied to an I2C touch controller to when the I2C controller is responding to Human Interface Device (HID) commands and providing touch reports must not exceed 100 milliseconds.

Additional Information

Business Justification	Windows 8 will support extremely fast resume on systems that support connected standby. As touch is the primary input mechanism on connected standby capable tablets, it is imperative that the I2C touch controller be available to respond to HID commands and provide input reports almost immediately upon resume to ensure a consistent user experience.
Enforcement Date	Mar. 01, 2012

System.Client.Tablet.RequiredHardwareButtons

Tablet and Convertible PC must have four hardware buttons

Target Feature	System.Client.Tablet
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Hardware buttons must generate an event when a button is pressed. Hardware buttons cannot launch any customized user interface elements. Buttons refers to buttons or keys.

Customized user interface elements include overlays, pop-up dialogs, toasts or other transitory user interface elements that appears over the Start screen, Windows Store apps, or desktop. They do not include Windows Store apps or desktop apps. Examples:

- Showing customized overlay or a toast in response to the changed state of a keyboard key (for example: Caps Lock, Num Lock).

- Showing a customized overlay or a toast in response to the changed state of a touchpad.

The following user interactions are permitted:

- Using a designated button or hot key to launch a default mail client
- Using a designated button or hotkey to launch a default browser
- Using a designated button or hotkey to launch a desktop application
- Using a designated button or launch a Windows Store app associated with a file or protocol handler
- Showing a notification in response to a hardware error (for example: a problem docking or undocking the system)

The following buttons must be implemented on a tablet or convertible PC:

Button	Function
Power	For Connected Standby machines, Power held for less than 2 seconds places the system into standby mode; press and hold for between two to ten seconds displays the "Slide to Shutdown" UI. Press and hold for 10 or greater seconds shuts down the system. If the machine does not support Connected Standby, Power Places the system into standby mode; press and hold for four seconds shuts down the system.
Windows Key Button	Navigates to the Start Screen
Volume up and down	Controls the audio volume in the PC
Windows Key Button + Volume Up	Narrator starts automatically
Windows Key Button + Volume Down	Screen Capture
Windows Key Button + Power Button	Sends the Secure attention Sequence (SAS) signal for bringing up the secure log in screen (Ctrl+Alt+Del)

A screen rotation lock button can be implemented. The rotation lock button can either be a press button or a slider that is stateless as long as there is no mechanical position.

Volume Buttons

The volume control buttons must be a soft press button type. Volume may be implemented as a single physical button, but must present two distinct interrupts via two distinct GPIO resources for SOCs, one for volume up and one for volume down.

Power Button

The power button (and, if applicable, sleep button and lid switch devices) must be implemented using the ACPI control method button definition described in sections 4.7.2.2.1.2, 4.7.2.2.2, and 4.7.4.2.1 of Advanced Configuration and Power Interface Specification (http://www.acpi.info/spec.htm).

Windows Button

A Windows button is required on any battery powered touch-enabled system without integrated or permanently attached keyboard, this includes small single hand (< 10") tablets and large (< 17") "moveable" all-in-ones. The Windows button can only send the Windows key scan code. The Windows logo that goes on the button, orientation and style guide are documented in the Logo License Agreement.

For tablets and convertibles of all screen sizes, the Windows button can be placed anywhere on the tablet. The design should consider convenient placement for left and right-handed users. Swiping across the button must not interfere with Windows gestures.

If the tablet is in a docking station with a keyboard, it is acceptable that the Windows button is covered by the docking station.

If the system is a convertible, the buttons must be accessible in all configurations.

The Windows button toggles between the last app and the Start Screen.

The preference is that the Windows button must be a soft press or mechanical button type. It is acceptable to implement a capacitive button.

Additional Information

Business Justification	These hardware buttons enable a navigation experience in the absence of a keyboard. System builders should consider whether any type of dynamic feedback is required for any capacitive buttons for visually impaired users.
Enforcement Date	Jun. 26, 2013

System.Client.Tablet.Graphics

These requirements describe the graphics requirements for Tablet PCs.

Related Requirements	• System.Client.Tablet.Graphics.MinimumResolution • System.Client.Tablet.Graphics.SupportAllModeOrientations

System.Client.Tablet.Graphics.MinimumResolution

Tablet PCs support minimum resolution and color

Target Feature	System.Client.Tablet.Graphics
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The minimum native resolution/color depth is 1024x768 at a depth of 32bits. The physical dimensions of the display panel must match the aspect ratio the native resolution. The native resolution of the panel can be greater than 1024 (horizontally) and 768 (vertically).

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.Tablet.Graphics.SupportAllModeOrientations

Graphics drivers on Tablet systems are required to support all mode orientations

Target Feature	System.Client.Tablet.Graphics
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Graphics drivers on tablet systems are required to support all mode orientations for every resolution enumerated for the integrated panel.

- A graphics driver is required to enumerate source modes for the integrated display. For each source mode enumerated the graphics driver is required to support each orientation (0, 90, 180 and 270).
- Each orientation is required even if the integrated panel is in a duplicate or extended topology with another display device. For duplicate mode, it is acceptable to rotate all targets connected to the rotated source. Per path rotation is allowed but not required.

Both the above mentioned requirements are optional for Stereo 3D capable resolutions.

Additional Information

Business Justification	Windows 8 is designing key experiences that depend on the ability of a user to be able to rotate the physical device. For this experience, it is critical that the desktop also rotate to be in sync with the device. Therefore the graphics driver must support each orientation for each mode.
Enforcement Date	Jun. 26, 2013

System.Client.UMPC.Graphics

This section describes requirements for graphics devices within ultra mobile pc client systems.

Related Requirements	• System.Client.UMPC.Graphics.WDDM

System.Client.UMPC.Graphics.WDDM

Display subsystem meets minimum GPU, memory, and resolution requirements for a Basic Windows experience on Ultra Mobile PCs

Target Feature	System.Client.UMPC.Graphics
Applies to	• Windows 7 Client x86, x64

Description

For Windows 7, Ultra Mobile PCs are required to ship with at least Direct3D10 Feature Level 9_3 and WDDM v1.1

- Minimum display resolution for ultra-mobile implementations is 800x600. 32 MB of memory available for graphics, when native display resolution is configured to 1024x768 or less.
- A minimum color-depth of 32 bpp.

A DirectX9.L-class GPU that supports Pixel Shader 2.0.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.VideoPlayback

The requirements in this section detail the abilities of the graphics subsystem and its ability to playback high definition video content.

Related Requirements	• System.Client.VideoPlayback.GlitchfreeHDVideoPlayback • System.Client.VideoPlayback.GlitchfreePlayback • System.Client.VideoPlayback.WNGlitchfreeHDVideoPlayback

System.Client.VideoPlayback.GlitchfreeHDVideoPlayback

System is capable of playing protected and unprotected High-Definition content with no perceivable glitch during playback on both AC and DC power modes

Target Feature	System.Client.VideoPlayback
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT)

Description

Required

System is capable of playing back protected and unprotected high-definition video of the following profiles with no perceivable glitch and user experience elements:

- 720p WMV - Traditional Content
  - 720p WMV Video (7mbps at 16x9 aspect ratio), WMA Audio 192 kbps, 48khz, 2-chanel 16-bit
  - 1280x720 (720p) ~ 7Mbps at Aspect Ratio 16x9,Video: Windows Media Video 9 Advanced Profile, Audio: Windows Media Audio 9 Professional at 384 kbps, 44.1 kHz, 2-channel 24-bit (A/V) 2-pass CBR
- 720p H.264 - Local and Internet Streaming
  - 720p H.264 high profile, 5mbps, AAC Audio 192 kbps, 48khz, 2-chanel 24-bit
- 1080p H.264 - Local and Internet Streaming
  - 1080p H.264 high profile, 5mbps, AAC Audio 192 kbps, 48 kHz, 2-channel 24 bit
- For DXVA Video drivers, we expect 720p (H.264 and VC-1) core device creation + decoder creation time < 50ms.

Optional
System is capable of playing back high-definition video of the following profiles with no perceivable glitch:

- Premium TV Content
    - TV 1080i 12mbps
- Complex Camera Content
    - AVCHD 1080i 20-35 mbps, high frame rate
    - AAC Audio 192 kbps, 48 kHz, 2-channel 24 bit

Design Notes:
If the system has any of the following configurations:

- Multiple GPU's
- Multiple output connections
- It is powered via an AC adapter and used away from an outlet using a rechargeable battery for continuous operation when not plugged in

Then validation of this requirement must be done against each configuration that applies to the system.

Additional Information

Business Justification	Windows has and will continue to move further into non-traditional PC locations, such as the living room and other small form factor entertainment devices. This movement will drive Windows further into scenarios historically provided by dedicated consumer electronics devices. Consumers will increasingly expect their PCs to provide the quality & reliability typically provided by dedicated consumer electronics devices.
Enforcement Date	Mar. 01, 2012

System.Client.VideoPlayback.GlitchfreePlayback

System is capable of playing High-Definition content with no perceivable glitch during playback

Target Feature	System.Client.VideoPlayback
Applies to	• Windows 7 Client x86, x64

Description

System is capable of playing back high-definition video of the following profile with no perceivable glitch:

- 1280x720 (720p) ~ 7Mbps at Aspect Ratio 16x9
- Audio: Windows Media Audio 9 Professional at 384 kbps, 44 kHz, 2-channel 24-bit (A/V) 2-pass CBR

- Video: Windows Media Video 9 Advanced Profile

Additional Information

Business Justification	Windows has and will continue to move further into non-traditional PC locations, such as the living room and other small form factor entertainment devices. This movement will drive Windows further into scenarios historically provided by dedicated consumer electronics devices. Consumers will increasingly expect their PCs to provide the quality & reliability typically provided by dedicated consumer electronics devices. In order to provide compelling multimedia experiences, it is critical for Windows to meet & exceed the digital media experiences provided by current and future generation consumer electronics devices. In addition, user feedback has shown the legacy multimedia capabilities of Windows have not provided the quality multimedia processing that user's expect.
Enforcement Date	Mar. 01, 2012

System.Client.VideoPlayback.WNGlitchfreeHDVideoPlayback

System is capable of playing protected and unprotected High-Definition content with no perceivable glitch during playback on both AC and DC power modes.

Target Feature	System.Client.VideoPlayback
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Required

Video Playback
System is capable of playing back protected and unprotected high-definition video – with captions enabled and disabled – for the following profiles with no audible or perceivable glitch. An audio or video glitch is measured through event trace from Windows instrumentation that are fired by the operating system:

- VC1
    0. Video: 1080p VC1, 18 Mbps, 30 FPS
    1. Audio: WMA Pro, 2 channels, 128 Kbps, 44.1 KHz
- H.264 TV
    2. Video: 1080p H.264, 18 Mbps, 30 FPS
    3. Audio: AAC, 2 channel, 128 Kbps, 44.1 KHz
- H.264 Film
    4. Video: 1080p H.264, 18 Mbps, 24 FPS
    5. Audio: Dolby Digital Plus, 384 Kbps, 6 channel, 48 KHz

- DirectX Core device creation + decoder creation time < 100ms for all required test content above.

Audio playback

System is capable of playing back the following protected and unprotected audio content profiles without audible glitches. This requirement also applies when system is in connected standby (CS) state, if implemented, and with audio offload enabled and disabled, if implemented.

- Audio: AAC, 192 Kbps, 2 channel, 48 KHz
- Audio: AAC, 192 Kbps, 2 channel, 44.1KHz

Optional

System is capable of playing back high-definition video of the following profiles with no perceivable glitch:

- Premium TV Content
    a. TV 1080i 12mbps
- Complex Camera Content
    b. AVCHD 1080i 20-35 mbps, high frame rate
    c. AAC Audio 192 kbps, 48 kHz, 2-channel 24 bit

Additional Information

Business Justification	Windows has and will continue to move further into non-traditional PC locations, such as the living room and other small form factor entertainment devices. This movement will drive Windows further into scenarios historically provided by dedicated consumer electronics devices. Consumers will increasingly expect their PCs to provide the quality & reliability typically provided by dedicated consumer electronics devices.
Enforcement Date	Jun. 26, 2013

System.Client.Webcam

These requirements apply to cameras integrated into the system.

Related Requirements	• System.Client.Webcam.Device • System.Client.Webcam.PhysicalLocation • System.Client.Webcam.VideoCaptureAndCamera

System.Client.Webcam.Device

Systems with camera must meet camera device requirements

Target Feature	System.Client.Webcam
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

If system has camera, it must meet the quality and requirements specified in the Device.Streaming.Webcam requirements (when applicable), on the system seeking certification.

- Device.Streaming.Webcam.Base.AVStreamWDMAndInterfaceRequirements
- Device.Streaming.Webcam.Base.BasicPerf
- Device.Streaming.Webcam.Base.DirectShowAndMediaFoundation
- Device.Streaming.Webcam.Base.KSCategoryVideoCameraRegistration
- Device.Streaming.Webcam.Base.MultipleClientAppSupport
- Device.Streaming.Webcam.Base.SurpriseRemoval
- Device.Streaming.Webcam.Base.UsageIndicator
- Device.Streaming.Webcam.H264.H264Support
- Device.Streaming.Webcam.NonMSDriver.VideoInfoHeader2
- Device.Streaming.Webcam.USBClassDriver.UVC
- Device.Streaming.Webcam.USBClassDriver.UVCDriver

Note: If there are multiple cameras, one indicator is acceptable so long as all cameras can control the indicator.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.Webcam.PhysicalLocation

Systems with a camera must report the physical location of the camera

Target Feature	System.Client.Webcam
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

For any camera device that is built into the chassis of the system and has mechanically fixed direction, the firmware must provide the _PLD method and set the panel field (bits[69:67]) to the appropriate value for the panel on which the camera is mounted. For example, "Front" indicates the camera view the user (webcam), while "back" indicates that the camera views away from the end user (still or video camera).

In addition, bit 143:128 (Vertical Offset), and bits 159:144 (Horizontal Offset) must provide the relative location of the camera with respect to the display. This origin is relative to the native pixel addressing in the display component. The origin is the lower left hand corner of the display, where positive Horizontal and Vertical Offset values are to the right and up, respectively. For more information see the ACPI version 5.0 Section 6.1.8 "Device Configuration _PLD (Physical Device Location)".

All other fields in the _PLD are optional.

Additional Information

Business Justification	This enables developers the ability to write applications that take advantage of front or rear facing cameras.
Enforcement Date	Mar. 01, 2012

System.Client.Webcam.VideoCaptureAndCamera

Video capture and cameras meet requirements and can support Windows Capture Infrastructure

Target Feature	System.Client.Webcam
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Scenarios

Previewing through the camera locally (viewfinder)

Taking a photo

Recording a video

Real-time Communication

This requirement applies to systems with non-USB built-in cameras.

Driver Model

A camera driver must be provided. AVStream is the required driver model.

AVStream driver is recommended to be a child driver of the WDDM driver. (IHV graphics driver must report the child slot as a child of the graphics device in DxgkDdiQueryChildRelations with correct AcpiUid. Later IHV driver should report the correct connect status in DxgkDdiQueryChildStatus.

AVStream Camera Driver shall be installed through device driver which meets the Windows security requirement. The driver shall not crash or hang the OS, and shall disallow memory violations.

Driver shall support system memory.

Pins

The driver is required to expose the following pins types:

- Capture: Supporting PINNAME_VIDEO_CAPTURE pin category with NV12 format
- Image: Supporting PINNAME_IMAGE pin category with JPEG format (can be supported by IHV Plug-in Model) and an uncompressed format (at least one of the following uncompressed formats: RGB32 or NV12 – support for both formats are recommended for all resolutions). In addition, image pin must support software trigger, VideoControlFlag_Trigger.

Optionally, the driver may expose

- Preview: Supporting PINNAME_VIDEO_PREVIEW pin category with NV12 format. (This is the video streaming pin with lower priority and potentially lower resolution)

All of the pins must be able to

- Operate both independently and in combination without interfering with each other.
- All streaming pins must support progressive format. The driver should set the interlace flag on the media type and the sample correctly.

Performance

Recorded content using all media types exposed by the record pin must be glitch free. Glitch is defined as a discontinuity in the stream that exceeds +/- half of a single frame duration between frames at light conditions sufficient to achieve the maximum frame rate.

To achieve the maximum frame rate, where the resolution is the lesser of the maximum resolution exposed by the image pin or 8MP, the driver must be able to

- Time to capture and deliver first photo <= 2500 ms at light conditions
- Or time to capture and deliver first video frame <= 2000 ms at light conditions

Controls

If camera supports any of the following control, it must expose the capability through the associated property in Windows

- Basic Controls (synchronous)
    - Brightness
    - Contrast
    - Hue
    - White Balance
    - Backlight Compensation
    - Exposure Focus
    - Zoom
- Extended Controls
    - Extended Focus
    - Photo Mode (async)
        - Single Frame HDR
        - Photo Sequence
        - Normal
    - Torch mode
    - Flash State
    - Scene mode (async)
    - Face based controls
    - Region of interest
    - ISO Mode (async)
    - Field of View
    - Video stabilization

- Warm Start
- EV Compensation
- Trigger Time
- Thumbnails
- Camera Angle Offset

Video Stabilization

If available, latency due to Image Stabilization shall not be more than one frame buffering in the “low” setting, default should be off and should be controllable at the application level

Mirroring

The default state for mirroring must be "not mirrored".

Photo Sequence

If camera HW supports Photo Sequence, it must expose the capability through the Photo Mode property and comply with the performance requirements

Lag between the user take photo request and when the actual sampled frame being captured must be

- Less than 300ms at light conditions sufficient to achieve the maximum frame rate.

The latency is defined between that time when the photo is triggered and the sample to be delivered to the pipeline is captured (see performance).

Photo Sequence must be enabled by the device and driver to

- Support the same resolutions that are exposed in Normal mode
- Report the current frame rates possible in Photo Sequence Mode based on the current light conditions. Device must honor and not exceed the maximum frame rate set by the application.
- Support at minimum 4fps measured at lesser of the maximum resolution exposed by the image pin or 8MP.
- Provide at least 4 frames in the past at lesser of the maximum resolution exposed by the image pin or 8MP.
- Photo Sequence should be performed independently, regardless preview on/off.
- Provide frames continuously in Photo Sequence mode at lesser of the maximum resolution exposed by the image pin or 8MP.
- Support thumbnails, upon request, at 1/2x, 1/4x, 1/8x, and 1/16x of the width and height of the original image resolution.
- The JPEG image generated by the camera may optionally have EXIF metadata indicating the “flash fired” information. EXIF information shall not include personally identifiable information, such as location, unique ids, among others.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.Webcam.NMSD

Webcam features

Related Requirements	• System.Client.Webcam.NMSD.NonMSDriver

System.Client.Webcam.NMSD.NonMSDriver

Systems with USB camera which are not using the inbox class driver must meet camera device requirements

Target Feature	System.Client.Webcam.NMSD
Applies to	• Windows 8.1 Client x86, x64

Description

If system has camera, it must meet the quality and requirements specified in the Device.Streaming.Webcam requirements (when applicable), on the system seeking certification.

- Device.Streaming.Webcam.Base.AVStreamWDMAndInterfaceRequirements
- Device.Streaming.Webcam.Base.BasicPerf
- Device.Streaming.Webcam.Base.DirectShowAndMediaFoundation
- Device.Streaming.Webcam.Base.KSCategoryVideoCameraRegistration
- Device.Streaming.Webcam.Base.MultipleClientAppSupport
- Device.Streaming.Webcam.Base.SurpriseRemoval
- Device.Streaming.Webcam.Base.UsageIndicator
- Device.Streaming.Webcam.H264.H264Support
- Device.Streaming.Webcam.NonMSDriver.VideoInfoHeader2

Additional Information

Enforcement Date	Jun. 26, 2013

System.Client.Webcam.Specification

Video capture devices are cameras and audio/video input devices that bring streaming data into a PC. Cameras that ware included in this section are those that take live streaming audio/video and individual still images and either save the stream to the systems storage or to another computer.

Related Requirements	• System.Client.Webcam.Specification.CameraRequirements

System.Client.Webcam.Specification.CameraRequirements

Video capture and cameras must qualify and processing performance requirements

Target Feature	System.Client.Webcam.Specification

Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

If a front or rear facing webcam is integrated in the system, then the following requirements apply:

Feature	Specification
Frame Rate	15 FPS @ 200 lux

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.DebugPort

The ability to debug a system is crucial to supporting customers in the field and root-causing behavior in the kernel. Requirements in this area support the ability to kernel debug a Windows system.

Related Requirements	• System.Fundamentals.DebugPort.SystemExposesDebugInterface

System.Fundamentals.DebugPort.SystemExposesDebugInterface

System exposes debug interface that complies with Debug Port Specification

Target Feature	System.Fundamentals.DebugPort
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

The next version of Windows will support several different debug transports. They are listed below in the preferred order of implementation.
Hardware Debugging Transports

- Ethernet Network Interface Card from the supported list: http://msdn.microsoft.com/en-us/library/windows/hardware/hh830880
- USB 3.0 - xHCI controller compliant to xHCI debug specification.
- 1394 OHCI compliant Firewire controllers.
- USB2 OTG (on supported hardware for Windows, recommend XHCI debug instead).
- USB 2.0 EHCI debug (the debug enabled port must be user accessible).

- Legacy Serial (16550 compatible programming interface).

ADDITIONAL REQUIREMENTS
FOR ALL OF THE ABOVE IMPLEMENTATIONS THE FOLLOWING MUST APPLY:

- There must be at least one user accessible debug port on the machine. It is acceptable on systems which choose to not expose a USB port or any other acceptable port from the list above to instead require a separate debugging board or device that provides the ability to debug via one (or more) of the transports above. That device/board must terminate in the same standard port as would be used for the transport if it were 'onboard' the machine. If this device is required it must be documented in the system specifications, be user serviceable, be user installable on the machine, and available for sale from the machine's vendor.
- On retail PC platforms, it is strongly recommended that machines have 2 user accessible debug ports from the above list. The secondary debug port is required to debug scenarios where the first debug port is in use as part of the scenario. Microsoft is not responsible for debugging or servicing issues which cannot be debugged on the retail platform, or reproduced on development platforms.
- SoC development or prototype platforms provided to Microsoft for evaluation must have a dedicated debug port available for debugging. If the debug port is used for any scenarios that are expected to also be used on retail shipping devices, in that case, there must be a secondary debug port available for debugging. This is to ensure that SoC development platforms can be used to test and debug all scenarios for all available transports, including USB host and function.
- All debug device registers must be memory or I/O mapped. For example, the debug device must not be connected behind a shared bus such as SPI or I2C. This would prevent other devices on the same bus from being debugged.
- When enabled, the debug device shall be powered and clocked by the UEFI firmware during preboot, before transferring control to the boot block.

For additional information, see http://go.microsoft.com/fwlink/?LinkId=237141

Additional Information

Enforcement Date	Jun. 01, 2006

System.Fundamentals.DebugPort.USB

The ability to debug a USB3 system is crucial to supporting customers in the field and root-causing behavior in the kernel. Requirements in this area support the debugging capability for the xHCI controller based systems via a debug registers. Every system that has xHCI controller and USB3 external port should support via this port.

Related Requirements	• System.Fundamentals.DebugPort.USB.SystemExposesDebugInterfaceUsb

System.Fundamentals.DebugPort.USB.SystemExposesDebugInterfaceUsb

USB 3 system exposes debug interface that complies with Debug Port Specification

Target Feature	System.Fundamentals.DebugPort.USB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

Systems that support USB 3 are required to have xHCI controller(s) compliant to the xHCI debug specification. The xHCI controller(s) shall be memory mapped.

- There must be at least one user accessible USB 3.0 debug port on the machine.
- All USB 3.0 ports must protect against a short on the VBus pin such that if another USB host is connected, the USB circuitry is not damaged.
- USB 3.0 hubs must not be integrated into the SoC or PCH/Southbridge.

For additional information, see http://go.microsoft.com/fwlink/?LinkId=58376

Additional Information

Business Justification	The goal of this requirement is to ensure systems are debug-capable in the scenario in which USB3 devices are being used.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware

This feature includes requirements specific to system firmware.

Related Requirements	• System.Fundamentals.Firmware.ACPI • System.Fundamentals.Firmware.ACPIRequired • System.Fundamentals.Firmware.FirmwareSupportsBootingFromDVDDevice • System.Fundamentals.Firmware.FirmwareSupportsUSBDevices • System.Fundamentals.Firmware.HardwareMemoryReservation • System.Fundamentals.Firmware.NoExternalDMAOnBoot • System.Fundamentals.Firmware.UEFIBitLocker • System.Fundamentals.Firmware.UEFIBootEntries • System.Fundamentals.Firmware.UEFICompatibility • System.Fundamentals.Firmware.UEFIDefaultBoot • System.Fundamentals.Firmware.UEFILegacyFallback • System.Fundamentals.Firmware.UEFISecureBoot • System.Fundamentals.Firmware.UEFITimingClass

	• System.Fundamentals.Firmware.Update

System.Fundamentals.Firmware.ACPI

ACPI System Requirements

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

All systems must meet the following ACPI table requirements.

ACPI Table Requirements	
Root System Description Pointer (RSDP)	Required
Root or Extended System Description Table (RSDT or XSDT)	Required
Fixed ACPI Description Table (FADT)	Revision 5 is required for Hardware-reduced ACPI platforms and systems that support connected standby platforms
Multiple APIC Description Table (MADT)	Required
Core System Resources Description (CSRT)	Required for ARM systems if non-Standard timers or any shared DMA controllers are exposed to the OS
Debug Port Table (DBGP)	Required. DBG2 table is required instead for Hardware-reduced ACPI platforms and systems that support connected standby platforms.
Differentiated System Description Table (DSDT)	Required

DSDT Requirements

As per ACPI 4.0a, all devices in the ACPI namespace must include:

- A vendor-assigned, ACPI-compliant Hardware ID (_HID object).
- A set of resources consumed (_CRS object).

In addition, the following conditional requirements apply:

- If any devices in the namespace share the same Hardware ID, then each is required to have a distinct Unique Identifier (_UID object).
- If any device in the namespace is enumerated by its parent bus (Plug and Play buses), the address of the device on its parent bus (_ADR object) is required.
- If any device in the namespace is compatible with a Microsoft-provided driver, the Compatible ID (_CID object) defined for that device type is required.

General-Purpose Input/Output (GPIO) on an System that Supports Connected Standby

GPIO Controllers for pins used by Windows drivers or ASL control methods must appear as devices in the ACPI namespace.

Devices in the namespace that are connected to GPIO pins on an enumerated controller device must:

- Include GPIO IO Connection resource descriptors in their _CRS for any GPIO I/O pins connected. Include GPIO Interrupt Connection resource descriptors in their _CRS object for any GPIO interrupt pins connected.

Simple Peripheral Bus (SPB) on an System that supports Connected Standby

SPB Controllers for connections used by Windows drivers or ASL control methods must appear as devices in the ACPI namespace.

Devices in the namespace that are connected to an enumerated SPB controller device (UART, I2C, SPI) must include SPB Connection resource descriptors in their _CRS for the SPB Connection(s) used.

Power Button

The power button, whether implemented as an ACPI Control Method Power Button or as part of the Windows-compatible Button Array, must:

- Be able to cause the system to power-up when required.
- Generate the Power Button Override Event (Section 4.7.2.2.1.3 of the ACPI 4.0a specification) when held down for 4 seconds.

Control Method Power Button

Systems dependent on built-in (or connected) keyboards/mice for input must conform to the ACPI Control Method Power Button (Section 4.7.2.2.1.2 of the ACPI 4.0a Specification). In addition, systems that support connected standby must:

- Implement the ACPI Control Method Power Button (Section 4.7.2.2.1.2 of the ACPI 4.0a Specification) using a dedicated GPIO interrupt pin to signal button press events.
- Configure the power button's GPIO interrupt pin as a non-shared, wake-capable (ExclusiveAndWake) GPIO interrupt connection resource.
- List the Power Button's GPIO interrupt connection resource in the ACPI Event Information (_AEI object) of the GPIO controller device to which it is connected.
- Provide the event method (_Lxx or _Exx object) for the power button event under the GPIO controller device in the ACPI namespace.

NOTE: For systems that require a separate driver to handle power button presses, it is acceptable to have that

driver evaluate a control method that performs a Notify() on the Control Method Power Button device instead of using the GPIO-based solution above.

Button Array-based Power Button

Touch-first (keyboard-less) systems must:

- Implement the Windows-compatible Button Array device.
- Connect the power button to a dedicated GPIO interrupt pin.
- Configure the power button's GPIO interrupt pin as a non-shared, wake-capable (ExclusiveAndWake), Edge-triggered (Edge) GPIO interrupt connection resource, capable of interrupting on both edges (ActiveBoth).
- List the power button's GPIO Interrupt connection resource first in the Button Array device's _CRS object.

NOTE: For systems that require a separate driver to handle power button presses, it is acceptable to have that driver call the 5-Button array driver's power button event interface instead of using the GPIO-based solution above.

Time and Alarm Device

All battery-powered systems which are not capable of supporting Connected Standby are required to implement the Alarm capabilities of the ACPI Time and Alarm control method device.
Any system that supports Connected Standby that sets the "CMOS RTC Not Present" bit in the IAPC_BOOT_ARCH flags field of the FADT must implement the device's Time capabilities.

Additional Information

Business Justification	Devices and buses that are on the system must have an ACPI namespace to ensure compatibility with Windows.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware.ACPIRequired

System that support connected standby must contain required ACPI elements

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Systems that support connected standby contain a hardware abstraction layer (HAL) which is distinct from typical PCs. Connected Standby systems also routinely include busses and devices which are not Plug and Play compatible. In these cases, systems that support Connected Standby must include ACPI tables which conform to

the Microsoft Windows 8 ACPI Specification for the following devices:

- USB
- GPIO
- Simple peripheral busses
- Human proximity sensor
- Ambient light sensor
- Storage and boot
- Core system resources
- Power firmware
- Boot firmware
- Runtime firmware

Additional Information

Business Justification	Devices and buses that are on the system must have an ACPI namespace to ensure compatibility with Windows.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware.FirmwareSupportsBootingFromDVDDevice

System firmware supports booting from DVD device as defined by the El Torito specification

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

The system firmware must support booting the system DVD. The system firmware or option ROM must support the No-Emulation mode in the "El Torito" Bootable CD-ROM Format Specification, Version 1.0, for installing Windows® from optical media, such as bootable DVD media. The primary optical device must be bootable. This requirement applies to the primary optical storage and the primary bus to which the device is attached.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Fundamentals.Firmware.FirmwareSupportsUSBDevices

System firmware provides USB boot support for USB keyboards, mouse, and hubs

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

The system firmware must: Support USB keyboards and pointing devices during system boot, resume from hibernate, and operating system setup and installation. Support USB input devices at least two levels of physical hubs below the host controller. Support composite input devices by the boot protocol as defined in HID. For Windows 8 systems, it is acceptable to enumerate, but not initialize all devices. If the device is accessed, it must be fully initialize before proceeding.

The USB controller and USB devices must be fully enumerated when:

- Anything other than the Windows Boot Manager is at the top of the system boot order
- A boot next variable has been set to boot to something other than the Windows Boot Manager
- On a system where the Windows Boot Manager is at the top of the list, an error case has been hit, such that the firmware fails over from the Windows Boot Manager to the next item in the list
- Resuming from hibernate, if the system was hibernated when booted from USB
- Firmware Setup is accessed.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Fundamentals.Firmware.HardwareMemoryReservation

System.Fundamentals.Firmware.HardwareMemoryReservation

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

This requirement limits the amount of memory that is reserved by the hardware (including drivers or firmware) and not available to the OS or user applications on a system. Taking into consideration the changes required to meet this requirement, it will be introduced in a phased manner. During the 2013 timeframe, the following interim thresholds will apply

- <=2GB systems – max of 3% of 2GB (61.5MB)
- 2-3GB systems – max of 3% of 3GB (92.2MB)
- For all other systems, a max of 120MB
- If screen resolution exceeds 1366x768, an additional 8 bytes per pixel will be allowed. Note: The budgets above are intended to cover 2 full screen video memory reservations for graphics drivers at 1366x768 at 32 bytes per pixel – 8MB. The adjustment above takes into consideration machines with higher resolutions.

RAM Size	Screen resolution	Threshold
1GB	1366x768	61.5MB
2GB	1366x768	61.5MB
2GB	1920x1080	61.5MB + 8MB
3GB	1366x768	92.2MB
3GB	1920x1080	92.2MB + 8MB
4GB	Any	120MB

Applies to Windows Client OS SKUs only [including Windows RT]

Design Notes:

- Hardware memory reservation is computed as the difference between the physical memory that is mapped as visible to the Windows OS (excluding all device/firmware reservations) compared to the installed RAM on the machine
- Installed memory is queried via Query GetPhysicallyInstalledSystemMemory() and OS visible memory is queried via GlobalMemoryStatusEx() - ullTotalPhys

Post 2013, the following thresholds will apply:

- <=2GB systems – max of 2% of 2GB (41MB)
- 2-3GB systems – max of 2% of 3GB (61.5MB)
- For all other systems, a max of 120MB

If screen resolution exceeds 1366x768, an additional 8 bytes per pixel will be allowed as described above.

Additional Information

Business Justification	Hardware based memory reservations done early in boot limit the amount of visible physical memory that is not available for the Operating system or user applications throughout the lifetime of a machine and is especially significant for memory constrained devices.
Enforcement Date	Jun. 26, 2013

System.Fundamentals.Firmware.NoExternalDMAOnBoot

All external DMA ports must be off by default until the OS explicitly powers them through related controller(s).

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The firmware must protect physical memory from unauthorized internal DMA (e.g. GPU accessing memory outside of video-specific memory) and all unauthorized DMA-capable external ports prior to boot, during boot, and until such time as the OS powers up DMA ports via related bridge controllers. When the device enters a low-power state, DMA port device context must be saved, and restored upon returning to active state.

If the firmware has an option to enable and disable this protection, the shipping configuration must be with protection enabled, and turning protection off must be protected, for example with platform authentication via BIOS password.

Note that this requirement precludes the use of attached storage as boot media if it can only be accessed via an external DMA-capable port.

Additional Information

Business Justification	Required to protect BitLocker keys and other secrets in memory from disclosure to an attacker.
Enforcement Date	Jun. 26, 2013

System.Fundamentals.Firmware.UEFIBitLocker

A system with TPM that supports wired LAN in pre-OS must support the UEFI 2.3.1 EFI_DHCP4_PROTOCOL protocol and the UEFI 2.3.1 EFI_DHCP6_PROTOCOL (and the corresponding service binding protocols)

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

Systems which support TPM and wired LAN networking must support EFI_DHCP4_protocol, EFI_DHCP4_SERVICE_BINDING_PROTOCOL, EFI_DHCP6_protocol, and EFI_DHCP6_SERVICE_BINDING_PROTOCOL for wired LAN as defined in UEFI 2.3.1.

At pre-boot, BitLocker must be able to discover its Network Unlock provider on a Windows Deployment Server (WDS) via DHCP, and unlock the OS volume after retrieving a secret from WDS.

Details

All UEFI systems with TPM present and a wired LAN port must support BitLocker Network Unlock . This requires full DHCP support for wired LAN during preboot through a UEFI DHCP driver. Specifically, there must be UEFI driver implementations for EFI_DHCP4_protocol, EFI_DHCP4_SERVICE_BINDING_PROTOCOL , EFI_DHCP6_protocol, and EFI_DHCP6_SERVICE_BINDING_PROTOCOL for wired LAN, as defined in UEFI 2.3.1.

This requirement is "If Implemented" for Server systems and applies only if a Server system is UEFI capable

Additional Information

Exceptions	This requirement is exempt for systems that are configured with Wireless LAN only.
Business Justification	Windows features must work as expected when the requisite hardware is present on a Logo system. UEFI is the core prerequisite of the Windows 8 platform that enables key features in security and performance.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware.UEFIBootEntries

UEFI firmware honors software control over load option variables

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

UEFI systems must allow the Operating System to create both generic and device specific boot entries with Messaging Device path, specifically USB Class Device Path (UEFI version 2.3 main specification section 9.3.5.9). The firmware must respect these settings and not modify them once the OS has changed them. Furthermore, the firmware must accurately report the boot entries to the OS.
Functional Notes:
If the device corresponding to a boot entry is not found, it is preferable for the system to proceed to the next boot entry silently (i.e. without presenting an error message or requiring user intervention).

If the system is booted from an internal USB device and there is a USB class entry at the top of the boot order, the system should first attempt to boot from external USB devices before attempting internal USB boot devices.
Design Notes:
The UEFI specification requires that the software bootmanager be allowed to do the boot order programming (UEFI v. 2.3 Section 3.1.1 "Boot Manager Programming").
The firmware should interpret load options and device paths as specified in Section 9 "Protocols - Device Path Protocol".

The UEFI specification describes the variables that must be modifiable at runtime in Section 3.2, table 10.

The UEFI specification is available at http://www.UEFI.org.

This requirement is "If Implemented" for Server systems and applies only if a Server system is UEFI capable

Additional Information

Exceptions	If a system ships with a non-UEFI-compatible OS, this requirement does not apply.
Business Justification	Currently boot order modification requires the user to manually change the firmware

	boot order. This process varies dramatically between OEMs and models. Creating a standardized programmatic way to modify boot order empowers both Microsoft and third-party developers to enable new scenarios. This requirement is already part of the spec and Intel's (the main contributor to the UEFI board as well as the creator of the benchmark firmware implementation) UEFI implementation already complies with the majority of this requirement and has announced that they will shortly upgrade their firmware to fully comply with this requirement (they have a bug with generic entries and boot from USB).
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware.UEFICompatibility

System firmware must meet Windows Compatibility requirements

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

All systems which ship with a UEFI-compatible OS must be compatible with the following sections of the UEFI 2.3.1 specification:
2.3, 3.1, 4.3, 6.1 ~ 6.5, 7.1~7.5, 8.1, 8.2, 9.1, 9.5, 11.2 ~ 11.4, 11.8, 11.9, 12.4, 12.7, 12.8, 12.9, 18.5, 21.1, 21.3, 21.5, 27.1~27.8.

Additional guidance listed in "UEFI Support and Requirements: Microsoft Windows Server 2008" document (available at http://www.microsoft.com/whdc/system/platform/firmware/uefireg.mspx), if any, shall also be required.

All Windows 8 systems must boot in UEFI mode by default. Other requirements may add additional sections of compatibility to this list, but this is the baseline.

All systems, except servers, must be certified in UEFI mode without activating CSM. If a system is available with 32bit and/or 64bit UEFI, both configurations must be tested for certification. For server, certification in UEFI mode is only required if UEFI is implemented.

OEMs may ship with CSM mode activated and the enterprise or government customer's licensed OS selection when requested.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware.UEFIDefaultBoot

All client systems must be able to boot into UEFI boot mode and attempt to boot into this mode by default

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

The System firmware must be able to achieve UEFI mode boot by default. Such a system may also support fallback to legacy BIOS mode boot for deploying OS images which do not support UEFI, if the user explicitly selects that option in the pre-boot UEFI BIOS menu.
This requirement is "If Implemented" for Server systems.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware.UEFILegacyFallback

System firmware must not fall back to legacy BIOS mode without explicit user action

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

If the system ships with a UEFI-compatible OS, system firmware must be implemented as UEFI and it must be able to achieve UEFI boot mode by default. Such a system may also support fallback to legacy BIOS boot on systems with OS which do not support UEFI, but only if the user selects that option in a pre-boot firmware user interface. Legacy option ROMs also may not be loaded by default.
"Explicit User Action" means that end user (or in case of enterprise customer, the IT pro) must manually access the pre-boot firmware configuration screen and change the setting. It may not ship in the BIOS mode by default and programmatic methods which can be attacked by malware are not acceptable.
All systems with Class 2 UEFI must not fall back to legacy BIOS mode nor load legacy Option ROM's without explicit user action within the pre-boot UEFI configuration UI."
An OEM may not ship a 64 bit system which defaults to legacy BIOS or loads legacy option ROMs if that system ships with a UEFI-compatible OS.
When Secure Boot is Enabled, Compatibility Support Modules (CSM) must NOT be loaded. Compatibility Support Modules are always prohibited on systems that support connected standby.
This requirement is "If Implemented" for Server systems and applies only if a Server system is UEFI capable

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware.UEFISecureBoot

All client systems must support UEFI Secure boot

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

Note: These requirements are "If Implemented" for Server systems and apply only if a Server system supports UEFI Secure Boot.

1. For the purposes of UEFI Secure Boot, the platform shall expose an interface to Secure Boot, whereby the system firmware is compliant with the following sections and sub-sections of UEFI version 2.3.1 Errata C:
    a. 7.1
    b. 7.2
    c. 7.2.1
    d. 27.2
    e. 27.5 - 27.8 (as further profiled below).
2. **Secure Boot must ship enabled** Configure UEFI Version 2.3.1 Errata C variables SecureBoot=1 and SetupMode=0 with a signature database (EFI_IMAGE_SECURITY_DATABASE) necessary to boot the machine securely pre-provisioned, and include a PK that is set and a valid KEK database. The system uses this database to verify that only trusted code (for example: trusted signed boot loader) is initialized, and that any unsigned image or an image that is signed by an unauthorized publisher does not execute. The contents of the signature database is determined by the OEM, based on the required native and third-party UEFI drivers, respective recovery needs, and the OS Boot Loader installed on the machine. The following Microsoft-provided EFI_CERT_X509 signature shall be included in the signature database: "CN=Microsoft Windows Production PCA 2011" and "Cert Hash(sha1): 58 0a 6f 4c c4 e4 b6 69 b9 eb dc 1b 2b 3e 08 7b 80 d0 67 8d" which shall use the following SignatureOwner GUID: {77fa9abd-0359-4d32-bd60-28f4e78f784b}, must also be included in the form of either an EFI_CERT_X509_GUID or EFI_CERT_RSA2048_GUID type.
    a. Note: Must NOT contain the following certificate: "CN=Microsoft Windows PCA 2010" and "Cert Hash(sha1): c0 13 86 a9 07 49 64 04 f2 76 c3 c1 85 3a bf 4a 52 74 af 88"

b. Note II: Windows Server systems may ship with Secure Boot disabled, but all other provisions of this sub-requirement must be met

3. When Secure Boot is Enabled, Compatibility Support Modules (CSM) must NOT be loaded. Compatibility Support Modules are always prohibited on Connected Standby systems.

4. **The initial UEFI signature databases (db) shall be created with the EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS attribute stored in firmware flash** and may be updated only with an OEM-signed firmware update or through UEFI authenticated variable write.

5. **Support for the UEFI "forbidden" signature database** (EFI_IMAGE_SECURITY_DATABASE1) must be implemented.

6. The platform shall ship with an initial, possibly empty, "forbidden" signature database (EFI_IMAGE_SECURITY_DATABASE1) created with the EFI_VARIABLE_TIME_BASED_AUTHENTICATED_ACCESS attribute. When a signature is added to the forbidden signature database, upon reboot, any image certified with that signature must not be allowed to initialize/execute.

7. **Secure Boot must be rooted in a protected or ROM-based Public Key**. Secure Boot must be rooted in an RSA public key with a modulus size of at least 2048 bits, and either be based in unalterable ROM or otherwise protected from alteration by a secure firmware update process, as defined below.

8. **Secure firmware update process**. If the platform firmware is to be serviced, it must follow a secure update process. To ensure the lowest level code layer is not compromised, the platform must support a secure firmware update process that ensures only signed firmware components that can be verified using the signature database (and are not invalidated by the forbidden signature database) can be installed. UEFI Boot Services variables must be hardware-protected and preserved across flash updates. The Flash ROM that stores the UEFI BIOS code must be protected. Flash that is typically open at reset (to allow for authenticated firmware updates) must subsequently be locked before running any unauthorized code. The firmware update process must also protect against rolling back to insecure versions, or non-production versions that may disable secure boot or include non-production keys. A physically present user may however override the rollback protection manually. In such a scenario (where the rollback protection is overridden), the TPM must be cleared. Further, it is recommended that manufacturers writing BIOS code adhere to the NIST guidelines set out in NIST SP 800-147 (http://csrc.nist.gov/publications/nistpubs/800-147/NIST-SP800-147-April2011.pdf), BIOS Protection Guidelines, which provides guidelines for building features into the BIOS that help protect it from being modified or corrupted by attackers. For example, by using cryptographic digital signatures to authenticate BIOS updates.

9. **Signed Firmware Code Integrity Check**. Firmware that is installed by the OEM and is either read-only or protected by a secure firmware update process, as defined above, may be considered protected. Systems shall verify that all unprotected firmware components, UEFI drivers, and UEFI applications are signed using minimum RSA-2048 with SHA-256 (MD5 and SHA-1 are prohibited), and verify that UEFI applications and drivers that are not signed as per these requirements will fail to run (this is the default policy for acceptable signature algorithms). If an images signature is not found in the authorized database, or is found in the forbidden database, the image must not be started, and instead, information about it shall be placed in the Image Execution Information Table.

10. UEFI firmware and driver implementations must be resistant to malicious input from untrusted sources. Incomplete input validation may result in buffer overflows, integer and pointer corruption, memory overwrites, and other vulnerabilities, compromising the runtime integrity of authenticated UEFI components.

11. **Verify Signature of all Boot Apps and Boot Loaders**. Upon power-on, the platform shall start executing boot firmware and use public key cryptography as per algorithm policy to verify the signatures of all images in the boot sequence up-to and including the Windows Boot Manager.

12. **Microsoft Key Encryption Key (KEK) is provisioned** A valid Microsoft-provided KEK is included in the KEK database. Microsoft provides the KEK in the form of either an EFI_CERT_X509_GUID or EFI_CERT_RSA2048_GUID type signature. The Microsoft KEK signature uses the following SignatureOwner GUID: {77fa9abd-0359-4d32-bd60-28f4e78f784b}.

13. **PKpub verification**. The PKpub key is owned by the OEM and stored in firmware flash. The private-key counterpart to PKpub is PKpriv, which controls Secure Boot policy on all OEM-manufactured devices, and its protection and use must be secured against un-authorized use or disclosure. PKpub must exist and the operating system must be able to read the value and verify that it exists with proper key length.

14. **No in-line mechanism is provided whereby a user can bypass Secure Boot failures and boot anyway** Signature verification override during boot when Secure Boot is enabled is not allowed. A physically present user override is not permitted for UEFI images that fail signature verification during boot. If a user wants to boot an image that does not pass signature verification, they must explicitly disable Secure Boot on the target system.

15. **UEFI Shells and related applications**. UEFI Modules that are not required to boot the platform must not be signed by any production certificate stored in "db", as UEFI applications can weaken the security of Secure Boot. For example, this includes and is not limited to UEFI Shells as well as manufacturing, test, debug, RMA, & decommissioning tools. Running these tools and shells must require that a platform administrator disables Secure Boot.

16. **Secure Boot Variable**. The firmware shall implement the SecureBoot variable as documented in Section 3.2 "Globally Defined Variables' of UEFI Specification Version 2.3.1 Errata C"

17. On non-ARM systems, the platform MUST implement the ability for a physically present user to select between two Secure Boot modes in firmware setup: "Custom" and "Standard". Custom Mode allows for more flexibility as specified in the following:

    - A.It shall be possible for a physically present user to use the Custom Mode firmware setup option to modify the contents of the Secure Boot signature databases and the PK. This may be implemented by simply providing the option to clear all Secure Boot databases (PK, KEK, db, dbx), which puts the system into setup mode.
    - B.If the user ends up deleting the PK then, upon exiting the Custom Mode firmware setup, the system is operating in Setup Mode with SecureBoot turned off.
    - C.The firmware setup shall indicate if Secure Boot is turned on, and if it is operated in Standard or Custom Mode. The firmware setup must provide an option to return from Custom to Standard Mode which restores the factory defaults. On an ARM system, it is forbidden to enable Custom Mode. Only Standard Mode may be enabled.

18. **Enable/Disable Secure Boot**. On non-ARM systems, it is required to implement the ability to disable Secure Boot via firmware setup. A physically present user must be allowed to disable Secure Boot via firmware setup without possession of PKpriv. A Windows Server may also disable Secure Boot remotely using a strongly authenticated (preferably public-key based) out-of-band management connection, such as to a baseboard management controller or service processor. Programmatic disabling of Secure Boot either during Boot Services or after exiting EFI Boot Services MUST NOT be possible. Disabling Secure Boot must not be possible on ARM systems.

19. **If the firmware is reset to factory defaults, then any customized Secure Boot variables are also factory reset**. If the firmware settings are reset to factory defaults, all custom-set variables shall be erased and the OEM PKpub shall be re-established along with the original, manufacturer-provisioned signature databases.

20. **OEM mechanism exists to remediate failed EFI boot components up to and including the Windows OS loader (bootmgr.efi)**. Images in the EFI boot path that fail Secure Boot signature verification MUST not be executed, and the EFI_IMAGE_EXECUTION_INFO_TABLE entry for that component shall be updated with the reason for the failure. The UEFI boot manager shall initiate

recovery according to an OEM-specific strategy for all components up to and including Windows bootmgr.efi.

21. **A working Windows RE image must be present on all Windows 8 client systems** The Windows Recovery image must be present in the factory image on every Secure Boot capable system. To support automated recovery and provide a positive user experience on Secure Boot systems, the Windows RE image must be present and enabled by default. As part of the Windows 8 Trusted Boot work enhancements have been made to Windows RE to allow optimized recovery from signature verification failures in Secure Boot. OEMs must include Windows RE as part of their factory image on all Windows 8 client systems.

22. **Firmware-based backup and restore**. If the OEM provides a mechanism to backup boot critical files (for example: EFI drivers and boot applications), it must be in a secure location only accessible and serviceable by firmware. The OEM may provide the capacity via firmware or other backup store to store backup copies of boot critical files and recovery tools. If such a store is implemented, the solution must also have the capability to restore the target files onto the system without the need for external media or user intervention. This is a differentiator for the OEM in failover protection, used if the Windows OS loader (bootmgr.efi) or other boot critical components fail, preventing Windows native recovery solutions to execute.

23. **Firmware-based backup synchronization**. Backup copies of boot critical components (for example: EFI drivers and boot applications) stored in firmware must be serviced in sync with updates to same files on the system If the system has the capability to store a backup copy of the Windows OS loader (bootmgr.efi), and potentially other critical boot components, then the files must be serviced on the same schedule as their counterparts in use on the live system. If the Windows OS loader is updated by Windows Update, then the backup copy of bootmgr.efi stored in firmware must be updated on the next boot.

24. **All Windows 8 client systems must support a secondary boot path**. For all Windows 8 systems configured for Secure Boot, there must be an alternate boot path option that is followed by the firmware in the event that the primary Windows OS loader fails. The second boot path may point either to the default shadow copy installed by Windows to the system backup store (<EFI System Volume>\EFI\Boot\boot<platform>.efi), or to a copy stored by the OEM firmware-based mechanism. This alternate path could be a file in executable memory, or point to a firmware-based remediation process that rolls a copy out of the OEM predetermined backup store.

25. **All Windows 8 client systems must support a USB boot path for recovery purposes**. For all Windows 8 systems configured for Secure Boot, there is a last resort of booting from USB.

26. **Supporting GetVariable() for the EFI_IMAGE_SECURITY_DATABASE** (both authorized and forbidden signature database) and the SecureBoot variable.

27. **Supporting SetVariable() for the EFI_IMAGE_SECURITY_DATABASE** (both authorized and forbidden signature database), using an authorized KEK for authentication.

28. **Reserved Memory for Windows Secure Boot UEFI Variables**. A total of at least 64 KB of non-volatile NVRAM storage memory must be reserved for NV UEFI variables (authenticated and unauthenticated, BS and RT) used by UEFI Secure Boot and Windows, and the maximum supported variable size must be at least 32kB. There is no maximum NVRAM storage limit.

29. **During normal firmware updates the following must be preserved**:

    a. The Secure Boot state & configuration (PK, KEK, db, dbx, SetupMode, SecureBoot)

    b. All UEFI variables with VendorGuid {77fa9abd-0359-4d32-bd60-28f4e78f784b}

    c. A physically-present user who authenticates to the firmware may change, reset, or delete these values

30. **The platform shall support EFI variables** that are:

    a. accessible only during the boot services or also accessible in the runtime phase;

    b. non-volatile; and

    c. possible to update only after proper authorization, for example, being properly signed.

The platform must support EFI variables with any valid combination of the following UEFI 2.3.1 variable attributes set:

Copy

EFI_VARIABLE_NON_VOLATILE

EFI_VARIABLE_BOOTSERVICE_ACCESS

EFI_VARIABLE_RUNTIME_ACCESS

EFI_VARIABLE_AUTHENTICATED_WRITE_ACCESS

EFI_VARIABLE_APPEND_WRITE

EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS

Connected Standby systems must meet all of the requirements cited in both "system.fundamentals.firmware.uefisecureboot" and "system.fundamentals.firmware.uefisecureboot.connectedstandby".

The documents referenced in this document may be requested by contacting http://go.microsoft.com/fwlink/p/?LinkId=237130.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Fundamentals.Firmware.UEFITimingClass

System firmware must expose timing and class information

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

- During POST, the firmware shall measure its own timing and record the duration of post, rounded to the nearest mSec
- These timings shall measure tEnd of reset sequence (Timer value noted at beginning of BIOS initialization - typically at reset vector) Handoff to OS Loader

A proposal for the structure of a table to record the duration of post is in review by ACPI committee and will be published as soon as practical.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware.Update

System firmware must meet the requirements in order to support system and/or device firmware updates using firmware driver package.

Target Feature	System.Fundamentals.Firmware
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

The requirements are required on ARM systems and are if-implemented for other systems. These requirements must be met by any system that updates system and/or device firmware using the Windows firmware driver package mechanism.

- The ESRT table must define at least one firmware resource (ESRE) in the resource list, which must include a system firmware resource.
- Only one system firmware resource can be defined in the ESRT.
- No two resources in the ESRT table are permitted to have the same firmware class GUID.
- ESRE must provide appropriate status code including success or failed firmware update attempt, on the subsequent boot, to the OS.
- Firmware for every resource defined by the ESRT must be upgradable to a newer version
- Firmware version of a particular resource must not break compatibility with firmware versions of other resources.
- Firmware must provide the lowest supported firmware version using the field "LowestSupportedFirmwareVersion" in the ESRE table. Firmware must not allow rollback to any version lower than the lowest supported version. Whenever a security related update has successfully been made, this field must be updated to match the "FirmwareVersion" field in the ESRE. When the lowest firmware version does not match the current firmware version, firmware must allow rollbacks to any version between the current version and the lowest supported version (inclusive).
- Firmware must seamlessly recover from failed update attempts if it is not able to transfer control to the OS after an update is applied.

Additional Information

Business Justification	System and Device firmware updates will be applied uniformly across all products running windows 8 and beyond

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware.Boot

This section describes boot requirements for all client systems.

Related Requirements	• System.Fundamentals.Firmware.Boot.EitherGraphicsAdapter • System.Fundamentals.Firmware.Boot.SystemWithBootDeviceGreaterThan

System.Fundamentals.Firmware.Boot.EitherGraphicsAdapter

System firmware must be able to boot a system with onboard or integrated graphics and with multiple graphics adapters

Target Feature	System.Fundamentals.Firmware.Boot
Applies to	• Windows 7 Client x86, x64

Description

Systems with GPUs on the system board and mobile systems that can use a docking station with PCI slots must provide a means in the system firmware setup utility to compel the system to use the onboard graphics device to boot. This capability is required so the onboard graphics device can be used in a multiple-monitor configuration and for hot undocking a mobile system.

If the system includes PCI, AGP, or PCI Express expansion slots, the system firmware must be able to boot a system with multiple graphics adapters. The system BIOS must designate one device as the VGA device and disable VGA on all other adapters. A system with an integrated graphics chipset and one or more discrete graphics adapters must be able to disable the integrated graphics chipset if the integrated graphics chipset cannot function as a non-VGA chipset.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Fundamentals.Firmware.Boot.SystemWithBootDeviceGreaterThan

Systems with a boot device with a capacity greater than 2.2 terabytes must comply with requirements

Target Feature	System.Fundamentals.Firmware.Boot
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64

	• Windows Server 2012 x64

Description

Systems with a boot device with a capacity greater than 2.2 terabytes must comply with the following requirements:

- The system must be 64-bit.
- The system must comply with Extensible Firmware Interface (EFI) 1.10 on Intel Itanium systems, and with native Unified Extensible Firmware Interface (UEFI) 2.0 or later on x64 systems.
- The system must comply with Advanced Configuration and Power Interface (ACPI) Specification version 4.0. Specifically, the system must be able to support legacy or Operating System-directed configuration and Power Management (OSPM)/ACPI mode.

Additional Information

Business Justification	Disk drive vendors are ready to ship greater than 2.2 TB disks into the market in 2010. OEM vendors are also preparing greater than 2.2 TB boot solutions for their new system platforms. To protect Microsoft® partners' investment and provide a good user experience with a greater than 2.2 TB disk drive, Microsoft only supports a greater than 2.2 TB boot scenario in UEFI systems. This certification requirement will provide the general design guidance for system vendors and IHVs to develop hardware, firmware and drivers that will support these disks.
Enforcement Date	Dec. 01, 2010

System.Fundamentals.Firmware.CS

Connected standby systems have additional UEFI Secure Boot requirements.

Related Requirements	• System.Fundamentals.Firmware.CS.CryptoCapabilities • System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby

System.Fundamentals.Firmware.CS.CryptoCapabilities

System that support Connected Standby must include cryptographic capabilities to meet customer expectations on platform speed and performance

Target Feature	System.Fundamentals.Firmware.CS
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Since all components in the boot path as well as many performance-critical OS subsystems will invoke cryptographic functions, run-time performance of these functions is critical. The following requirements have

been drafted to help ensure sufficient cryptographic capabilities are in place to meet customer expectations on platform speed and performance

1. The platform must meet cryptographic performance requirements as stated in Table 1. The platform may meet these requirements through any combination of hardware or software. The following general remarks apply to all algorithms in Table 1:

    a. The platform must pass the Windows Hardware Certification Kit test "Storage Performance EMMC" with a "CPU Utilization Percentage" value of less than 20%. "CPU Utilization Percentage" is the average CPU usage over the duration of the test for the entire system under test.

    b. Target performance must be achieved in a multi-threaded test. The number of threads will be determined by querying the property named L"HardwareThreads" on the BCrypt provider through the CNG BCryptGetProperty interface. The provider is required to return a DWORD value in response. If the provider does not support this property, the test will run single-threaded.

    c. When cryptographic acceleration engines are used: Due to the overhead involved in dispatching requests to hardware acceleration engines, it is recommended that small requests be handled in software. Similarly, it is recommended that vendors consider using CPU-based cryptography to improve throughput when all cryptographic acceleration engines are fully utilized, idle capacity is available on the CPU, and the device is in a high-performance mode (such as when connected to AC power).

2. ARM based platforms must implement the EFI_HASH_PROTOCOL from UEFI Industry Group, Unified Extensible Firmware Interface Specification version 2.3.1 Errata B. The EFI_HASH_PROTOCOL implementation must be accessible from Windows pre-Operating System code (i.e. in the Boot Services phase of platform boot). Both the UEFI hash protocols EFI_HASH_ALGORITHM_SHA1_NOPAD_GUID and EFI_HASH_ALGORITHM_SHA256_NOPAD_GUID must be supported, and the implementation must support passing a Message at least 10 Mbytes long (Note: No padding must be applied at any point to the input data).

3. To make entropy generation capabilities available to Windows pre-Operating System code, the platform shall support the EFI_RNG_PROTOCOL for pre-Operating System read of at least 256 bits of entropy in a single call (i.e. 256 bits of full entropy from a source with security strength of at least 256 bits). The protocol definition can be found in Microsoft Corporation, "UEFI Entropy-Gathering Protocol,"[2].

4. All cryptographic capabilities in accordance with Table 1 shall be accessible from the runtime OS in kernel mode, through the interface specified in Microsoft Corporation, "BCrypt Profile for SoC Acceleration," [2] .

5. **OPTIONAL.** It is recommended that the platform's cryptographic capabilities also be accessible from the runtime OS in user mode, through the interface previously referenced in Requirement 4.

6. The OS interface library shall be implemented in such a way that when an unprivileged process is operating on a given key in a given context, it shall not be able to access the key material or perform key operations associated with other contexts.

7. **OPTIONAL**. It is highly recommended that the RNG capability of the platform be exposed through an OS entropy source through the interface specified in Microsoft Corporation, "BCrypt Profile for SoC Acceleration," previously referenced in Requirement 4.

8. **OPTIONAL.** (Applies when a cryptographic acceleration engine is used) It should be possible to maintain and perform cryptographic operations on at least three distinct symmetric keys or two symmetric keys and one asymmetric key simultaneously in the acceleration engine.

Table 1: Algorithm-specific requirements. The "Category" column classifies algorithms as mandatory to support at the software interface as per requirement 4 (M), or optional (O). Note that all algorithms that are accelerated in hardware must also be exposed through the software interface.

Algorithm	Category	Modes	Mandatory Supported Key Size(s)	Remarks
3-DES	O	ECB, CBC, CFB8	112, 168	
AES	M	ECB, CBC, CFB8, CFB128, CCM, CMAC, GCM, GMAC	128, 192, 256	Performance >= 60 MBytes/s for AES-128-CBC and AES-128-ECB encryption and decryption as measured at the CNG kernel-mode BCrypt interface when processing 32 kByte blocks.
	O	CTR, XTS, IAPM	128, 192, 256	
RSA	O	PKCS #1 v1.5, PSS, OAEP	512 to 16384 in 8-byte increments	Public key performance for 2048-bit keys (and public exponent F4 (0x10001)) when verifying PKCS#1v1.5 padded signatures, measured at the kernel mode BCrypt interface <=0.6 ms/verification.
ECC	O	ECDSA, ECDH	256	If implemented, must support Elliptic curve P-256 defined in National Institute for Standards and Technology, "Digital Signature Standard," FIPS 186-3, June 2009.
SHA-1	O			Performance >60 Mbytes/s as measured at the CNG kernel-mode BCrypt interface when processing 4kByte blocks.
HMAC-SHA1	O			
SHA-256	O			Performance >60 Mbytes/s as measured at the CNG kernel-mode BCrypt interface when processing 4kByte blocks.
HMAC SHA-256	O			
RNG	M	Entropy source with optional FIPS 800-90-based DRBG		Security strength must be at least 256 bits[1]. Note that exposing this functionality through the UEFI Entropy-Gathering Protocol is required (see Req 3) and exposing it as an OS entropy source is recommended (see Req 7).

[1]The Connected Standby vendor shall supply documentation indicating, and allowing Microsoft to estimate, the quality of the entropy source. The entropy shall be assessed using the min-entropy method of Appendix C of National Institute for Standards and Technology, "Recommendation for Random Number Generation using Deterministic Random Bit Generators," FIPS 800-90, March 2007 and must surpass or be equal to 256 bits before the runtime OS starts.

[2]This specification must be requested explicitly from Microsoft. To request the current version, please contact http://go.microsoft.com/fwlink/?LinkId=237130.

Additional Information

Business Justification	Core cryptographic functions are used in Windows to provide platform integrity as well as protection of user data. Therefore, in order to have a safe and usable platform, these cryptographic building blocks must achieve certain standards of security and performance.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby

All client systems that support Connected Standby must support UEFI Secure Boot

Target Feature	System.Fundamentals.Firmware.CS
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

1. Connected-Standby systems must meet all of the requirements cited in this section and under System.Fundamentals.Firmware.Uefisecureboot section.

2. **Boot Integrity**. Platform uses on-die ROM or One-Time Programmable (OTP) memory for storing initial boot code and initial public key (or hash of initial public key) used to provide boot integrity, and provides power-on reset logic to execute from on-die ROM or secure on-die SRAM.

3. Secure Boot launch of Windows 8 BootMgr must not require use of an Allowed DB entry other than the Microsoft-provided EFI_CERT_X509 signature with "CN=Microsoft Windows Production PCA 2011" and "Cert Hash(sha1): 58 0a 6f 4c c4 e4 b6 69 b9 eb dc 1b 2b 3e 08 7b 80 d0 67 8d.

4. On ARM, the UEFI Allowed database ("db") must not contain any other entry than the Microsoft-provided EFI_CERT_X509 signature with "CN=Microsoft Windows Production PCA 2011" and "Cert Hash(sha1): 58 0a 6f 4c c4 e4 b6 69 b9 eb dc 1b 2b 3e 08 7b 80 d0 67 8d.

5. The policy for acceptable signature algorithms (and padding schemes) shall be possible to update. The exact method for updating the policy is determined by each authority (for example: Microsoft determines policies for binaries it is responsible for; SOC vendor for firmware updates). It is recognized that the initial ROM code need not have an ability to update the initial signature scheme.

6. The platform shall maintain and enforce a policy with regards to signature authorities for firmware and pre-Operating System components; the policy (and hence the set of authorities) shall be possible to update. The update must happen either as a result of actions by a physically present authorized user or by providing a policy update signed by an existing authority authorized for this task. On ARM platforms,

the physical presence alone is not sufficient. Signature authority (db or KEK) updates must be authenticated on ARM platforms.

7. Upon power-on, the platform shall start executing read-only boot firmware stored on-die and use public key cryptography as per algorithm policy to verify the signatures of all images in the boot sequence up-to the Windows Boot Manager.

8. Protection of physical memory from unauthorized internal DMA (for example: GPU accessing memory outside of video-specific memory) and all external DMA access to the SOC. The firmware shall enable this protection as early as feasible, preferably within the initial boot firmware.

9. **Optional**.The memory containing the initial boot firmware (executing in SRAM) may be made inaccessible upon jumping to the next validated stage of the boot sequence. The initial boot firmware may remain inaccessible until power-on-reset is triggered.

10. The platform shall enforce policy regarding the replacement of firmware components. The policy must include protection against rollback. It is left to the platform vendor to define the exact method for policy enforcement, but the signature verification of all firmware updates must pass and the update must be identified in such a manner that a later version of a component cannot, without proper authorization (for example: physical presence), be replaced by an earlier version of the component where earlier and later may be defined by a (signed) version number, for example.

11. **Optional**. The platform shall offer at least 112 logical eFuse bits to support platform firmware revision control in accordance with the above requirement.

12. Physical Security Requirements. In retail parts, once the platform is configured for Production mode, the hardware must disable all external hardware debug interfaces such as JTAG that may be used to modify the platform's security state, and disable all hardware test modes and disable all scan chains. The disabling must be permanent unless re-enablement unconditionally causes all device-managed keys that impact secure boot, TPM, and storage security to be rendered permanently erased.

13. On ARM platforms Secure Boot Custom Mode is not allowed. A physically present user cannot override Secure Boot authenticated variables (for example: PK, KEK, db, dbx).

14. Platforms shall be UEFI Class Three (see UEFI Industry Group, Evaluating UEFI using Commercially Available Platforms and Solutions, version 0.3, for a definition) with no Compatibility Support Module installed or installable. BIOS emulation and legacy PC/AT boot must be disabled.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Fundamentals.Firmware.CS.UEFISecureBoot

Connected standby systems have additional UEFI Secure Boot requirements.

Related Requirements	• System.Fundamentals.Firmware.CS.UEFISecureBoot.Provisioning

System.Fundamentals.Firmware.CS.UEFISecureBoot.Provisioning

Systems are required to leave manufacturing provisioned with all cryptographic seeds and keys that are necessary to prevent attachs

Target Feature	System.Fundamentals.Firmware.CS.UEFISecureBoot
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64

Description

Each device is required to leave manufacturing provisioned with all cryptographic seeds and keys that are necessary to prevent attacks against the device's Secure Boot, TPM and secure persistent storage implementations. Seeds and symmetric keys shall be immutable, per-device-unique, and non-predictable (random with sufficient length to resist exhaustive search; see NIST 800-31A for acceptable key sizes).
The platform is required to implement a hardware interface and share documentation and tools as specified in the 'Hardware Security Testability Specification' document (This document is available on Connect).
This requirement is IF IMPLEMENTED for Server.

Additional Information

Enforcement Date	Jan. 01, 2015

System.Fundamentals.Firmware.TPR

This feature includes requirements specific to system firmware with eDrive support.

Related Requirements	• System.Fundamentals.Firmware.TPR.UEFIEncryptedHDD

System.Fundamentals.Firmware.TPR.UEFIEncryptedHDD

Systems which ship with a self-encrypting hard drive as a storage device must support the UEFI 2.3.1 EFI_STORAGE_SECURITY_COMMAND_PROTOCOL protocols and shall contain a non-OS partition that can be used to store WinRE

Target Feature	System.Fundamentals.Firmware.TPR
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

If self-encrypted drive support is implemented it must have a UEFI-compatible OS and contain system firmware both conforming to system firmware logo requirements as defined in System.Fundamentals.Firmware and also contain a separate WINRE partition.

BitLocker shall support self-encrypting drivers that conform to the eDrive Device Guidelines available on WHDC at http://msdn.microsoft.com/en-us/library/windows/hardware/br259095

UEFI – Trusted Command Support in UEFI 2.3 + UEFI Mantis change number 616 or UEFI 2.3.1

Standard v2.3 + errata Bv2 www.uefi.org

Mantis Change Number 616 www.uefi.org (This is not part of v2.3)

All necessary partitions have to be created, managed individually pre/post encryption. The WINRE partition must always be separate and outside of the OS/encryption partition.

If WinRE is on the system partition, the size is 350MB. If it's not the system partition, then it's 300MB. This is assuming MBR layout. (For GPT, WinRE is always separate from the ESP, hence 300MB.)

This requirement is "If Implemented" for Server systems and applies only if a Server system is UEFI capable

Additional Information

Enforcement Date	Jun. 26, 2013

System.Fundamentals.Graphics

Base for Graphics on Systems

Related Requirements	• System.Fundamentals.Graphics.FirmwareSupportsLargeAperture • System.Fundamentals.Graphics.MicrosoftBasicDisplayDriver • System.Fundamentals.Graphics.MultipleOperatingMode • System.Fundamentals.Graphics.NoRebootUpgrade • System.Fundamentals.Graphics.PremiumContentPlayback • System.Fundamentals.Graphics.Windows7.MultipleOperatingModes

System.Fundamentals.Graphics.FirmwareSupportsLargeAperture

32-bit and 64-bit system firmware supports large aperture graphic adapters

Target Feature	System.Fundamentals.Graphics
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

The system firmware (BIOS/UEFI) must support large aperture graphics adapters. The 32-bit system firmware (BIOS/UEFI) must be able to support at least 256 MB aperture. On 64-bit systems the firmware (BIOS/UEFI) must be able to support at least 1GB aperture.

A system that supports multiple graphics adapters must ensure sufficient resources for each adapter. For example on a 32bit system with 4 graphics adapters, each adapter must receive at least 256MB memory resources each on the PCI bus.

Additional Information

Business Justification	The purpose of this requirement is to ensure that sufficient memory resources can be allocated on the PCI bus for large memory graphics adapters so that the system will successfully boot. This is to enable support for graphics adapters with larger size of memory that can be accessed over the bus by the host CPU.
Enforcement Date	Jun. 01, 2006

System.Fundamentals.Graphics.MicrosoftBasicDisplayDriver

System is compatible with the Microsoft Basic Display Driver

Target Feature	System.Fundamentals.Graphics
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

The System must boot in a mode where the frame buffer used by the Microsoft basic display driver is displayed whenever the Microsoft display driver writes to the frame buffer. No other driver is involved to accomplish this output. The frame buffer must be linear and in BGRA format.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Graphics.MultipleOperatingMode

If a Windows 8 system has Multiple GPU's, the graphics and system test must pass in every "Operating Mode"

Target Feature	System.Fundamentals.Graphics
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Multiple GPU features must meet all Graphics and System requirements and pass all tests in all available GPU operating modes.
Windows certification tests are designed to ensure supported core Windows experiences work correctly, the system is reliable and that developers can count on graphics APIs conforming to the specification. In a single GPU system, these features are validated by the graphics device tests and system tests.
Multiple GPU systems introduce special cases or modes that require special testing considerations. Multiple GPU's increases the complexity of the overall system and may expose the Windows system to unsupported features without careful analysis and testing.
For the purpose of certification, there are three types of GPU features:

1. Supported Features - these features were considered in the Windows design scenarios, are explicitly allowed, are tested by the Hardware Certification Kit, and can receive certification upon submission of the results.

2. Features that require an addendum to certify - these features do not have explicit support by Microsoft in the OS runtimes, API's or kernel, but may be certified by IHVs and used in certified systems if the conditions placed on their use are met by the IHVs and OEMs. The Windows OS is usually unaware of these features. However, Microsoft recognizes these features may be designed and supported by the IHV in a manner not contrary to the Windows scenarios, and if the partners commit to supporting these features over the lifetime of the system, Microsoft will not block their use. The Microsoft graphics technical representative can provide additional information when it is unclear if a feature outside the Windows supported designs would be certifiable with an addendum or not allowed.

3. Not allowed Features - these are features which create unavoidable scenarios running counter to Windows experience expectations, do not meet certification requirements and would prevent the system from getting certification.

Microsoft makes no guarantee that features other than Supported Features (type 1 above) work correctly, or will continue to work correctly after patches, service packs or OS upgrades. Such features may appear to work correctly under test conditions but may fail catastrophically at some point in the future. These changes can result in unpredictable behavior or application failures, particularly as the OS and applications evolve and expose discrepancies.
To assure no Windows features fail and to assure end-users that their experience will be reliable for the life of a product, the hardware manufacturers and the system manufacturers who are shipping a Windows feature that requires an addendum (type 2 above) must carry the responsibility for testing, supporting and servicing any divergence from supported features. To be certified, the device's additional quality assurance and future support will be documented using the addendum process.

Upon agreement of the terms of this Addendum the OEM:

1. Declares the intent to ship a feature that is not supported by Microsoft Windows

2. Declares no Windows features fail as a result of the feature.

3. Upon evidence of a passing submission of the system, the addendum may be extended to the system.

The conditions of the addendum are:

1. Company will, or will require the responsible IHV partner to, thoroughly test Affected Company Product(s) with the objective to proactively detect and correct potential issues, and in any event before any delivery of an update to the Affected Company Products.

2. Monitor the reliability of the Windows unsupported feature using telemetry including but not limited to telemetry data provided by Microsoft.

3. Upon discovery of a failure through telemetry or Microsoft notification, develop and timely distribute via Windows Update fixes for any end-user issues resulting in (1) Windows feature, application or kernel failures, (2) failures in third party Windows Store-style apps running on Windows, or (3) failures in Microsoft products including but not limited to Internet Explorer, Windows Live and Microsoft Office which are attributable to the Affected Company Product(s). Timely means within no more than 3 months after a failure is discovered and for (1) above, at least 3 months before a patch, a service pack or a future Windows operating system becomes available, for (2) above, at least 3 months before a fix or a new or updated release of the affected app becomes available, and for (3) above, at least 3 months before a patch, a service pack or a future release of the affected Microsoft products becomes available.

4. Company will enter into a separate agreement with the relevant Affected Company Product IHV to bind IHVs to the support obligations as described above.

5. Microsoft, in collaboration with the IHV, may continue to update and distribute a driver for the device beyond the addendum commitment.

The following guidelines detail some specific situations and typical problems this requirement is intended to address. This is not a comprehensive list of possible problematic situations. Other situations may be addressed in a successful addendum request for a specific Windows unsupported feature.

1. Multiple GPUs are not allowed on ARM platforms. Windows has chosen to not allow or support multiple GPU systems for the Windows 8 release on ARM based systems. Multiple GPU solutions would introduce additional complexity, and potential for untested features. For Windows 8 all ARM based systems must be configured with a single GPU only.

2. In features where multiple GPUs are allowed, the system must meet the minimum performance requirements in every mode that can be configured. Performance requirements are described in this requirement (*System.Fundamentals.Graphics.DisplayRender.Performance*). A system certification submission will need to include the system level performance related results with a submission for both GPUs.

3. Projection is a key Windows scenario, and the ability to clone the laptop lid to an external monitor with Win+P is a basic expectation of the end user. All mobile system must be able to "clone" or "extend" by using the Win+P key to the external port most likely used with a projector.

4. A system must correctly support the WDDM 1.2 "seamless boot" and bugcheck, PnP Start/Stop functionality:

    a. The "post" GPU must meet the seamless boot requirements as described in requirements Device.Graphics.WDDM12.Display.DisplayOutputControl, System.Client.Firmware.UEFI.GOP.Display, and System.Fundamentals.Graphics.InternalDisplay.NativeResolution

    b. PnP stop of the WDDM driver or a bugcheck must cause the frame buffer to be seamlessly handed off to the OS as described in requirement Device.Graphics.WDDM12.Display.PnpStopStartSupport

5. Microsoft tests and validates the graphics API's to ensure conformance with the D3D specification/MSDN documentation. The graphics features and capabilities are always available once they have been enumerated. Applications are designed with these expectations, that the GPU device and capabilities are fixed for the runtime of the application. A system where these expectations are not true is not allowed.

6. Windows supports surprise removals of the GPU only with hibernate. All other cases are not allowed. A system must not use a GPU that could be "surprise removed" in these other cases. An example of a feature that may result in unexpected surprise removal is a GPU in a docking station.

7. Heterogeneous multiple GPU solutions can be allowed, however systems shipping with heterogeneous devices must have been certified with all GPUs present. For system certification purposes heterogeneous should be considered a Windows unsupported feature, requiring an addendum.

8. Linked Display Adapter features have two operating modes: Linked, and Unlinked. Certification of devices that can be used in an LDA configuration must submit for certification with at least 1 device configured in the LDA feature and 1 device configured unlinked. This is to ensure that all expected Windows features pass in both modes.

9. Features involving an i-GPU with a secondary d-GPU require an addendum to be certified.

10. Lastly "Switchable Graphics solutions" where the display output is moved from one GPU to another GPU are not allowed.

The below table summarizes the above in a tabular format

GPU Feature Type	ARM platforms	All Other platforms
Single GPU	Windows Supported	Windows Supported
Homogeneous (Same Vendors)	Not Allowed	Windows Supported
Heterogeneous (Diff Vendors)	Not Allowed	Requires addendum
Linked Adapter	Not Allowed	Windows Supported
i- GPU with a Secondary d-GPU	Not Allowed	Requires addendum
Switchable Graphics	Not Allowed	Not Allowed

Design Notes:
The following are definitions of multiple GPU variations used in this requirement.
Homogeneous: A homogeneous multiple GPU feature consists of two or more GPUs from the same GPU vendor, using the same graphics driver. All GPUs are always on. Each GPU is visible to the application and there is no coordination between adapters and display output is not shared or switched between adapters.
Linked Adapter: A linked adapter (LDA) feature has two or more identical GPUs from the same vendor, which share the graphics workload. In an LDA feature an application will access one GPU, but the work is distributed by the driver across multiple GPUs. Typically the application is not even aware of the existence of the multiple GPU's. Examples are AMD CrossfireX and NVIDIA SLI.
Heterogeneous: A heterogeneous feature consists of two or more GPUs where there is no coordination between the adapters, and display output is not shared or switched between the adapters. The GPUs usually are from different GPU vendors although they may be from the same vendor. All GPUs are always on. In a heterogeneous feature each GPU is visible to the application.
i- GPU with a Secondary d-GPU: Integrated GPU (i-GPU) with a Secondary Discrete GPU (d-GPU) has the integrated GPU sharing computation and rendering with an additional discrete adapter. In this case the two GPUs may have different drivers and even be from different vendors. In this feature, displays are not switched between the adapters; the main display is always physically connected via the i-GPU. Examples are NVIDIA Optimus, AMD Hybrid PowerXpress and AMD Enduro.

Switch-able Graphics: A Switch-able Graphics feature is two or more GPUs from either the same vendor or different vendors where the responsibility for display output to any monitor changes from one processor to the other, typically via a MUX. This feature is not allowed in Windows 8 systems and above.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Graphics.NoRebootUpgrade

Graphics drivers must be upgradable without a reboot of the system

Target Feature	System.Fundamentals.Graphics
Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

The WDDM driver model has supported rebootless upgrade since Windows Vista. For Windows 8 all systems must support the upgrade of graphics driver package without requiring the system to reboot.

For example the graphics driver package includes the graphics driver and all associated utilities and services.

Additional Information

Business Justification	Windows 8 has removed XDDM which means the system is now always running a WDDM Driver. This means rebootless upgrade is now possible in all systems. This requirement is to ensure: A user is now able to accept a driver update without interrupting their activity. This provides a more modern experience. On server platforms reboots impact server availability.
Enforcement Date	Jun. 26, 2013

System.Fundamentals.Graphics.PremiumContentPlayback

Protected Environment Signature requirement

Target Feature	System.Fundamentals.Graphics
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Systems must be capable of premium (i.e., protected) audio or video content playback. If one or more (Audio Processing Objects) APOs are provided in the audio driver, the APOs must be able to load successfully without disabling the Audio Device Graph as a protected process.

Additional Information

Exceptions	This is if-implemented for Windows 8.1 Server as audio and video is not required components. If the server contains either an audio or full video component then this requirement is required.
Business Justification	Premium content playback using Windows Store apps require display driver and Audio Processing Object binaries to be properly signed as required by Windows Protected Environment license program. Playback fails if a required driver binary does not meet this requirement leading to poor user experience.
Enforcement Date	Jun. 26, 2013

System.Fundamentals.Graphics.Windows7.MultipleOperatingModes

If a Windows 7 system has Multiple GPU's, the graphics and system test must pass in every "Operating Mode"

Target Feature	System.Fundamentals.Graphics
Applies to	• Windows 7 Client x86, x64

Description

A system with multiple GPU's must meet logo requirements and pass testing in all available GPU modes.

Multi-GPU system may have one or more integrated and/or discrete display adapters.

A Hybrid/Switchable system must transition between GPU modes only on user consent. GPU mode transitions are triggered manually by the end-user unless it is completely transparent to the user with no visible effects. On initiation of the GPU switch, the Multiple GPU system must display an indication to the end-user to show which GPU mode they are currently in and running applications must not be stopped without user notification or consent. It is also possible to create a Pre-consent list by which the customer can chose to that the "dialog box" that came up during a mode switch be automatically for the particular scenario.

If a system running in any multi-GPU mode is connected to an external display, the system must be able to transition into a Clone mode or Extended Desktop mode through CCD. Recommended time for CCD mode change should not be longer than an equivalent desktop mode change- preferably less than 5 seconds. If the system is connected to an external display device through an output connector who's GPU was powered off, recommended time for GPU switching should not be longer than a preferred time of 10 seconds so the user doesn't think the system is hung or not functioning properly.
A multi-GPU system may have one or more integrated and/or one or more discrete display adapters. A multi-GPU system may have display adapters which do not support standard VGA. However, at least the boot device must be able to support standard VGA and the ability to reset to standard VGA resolutions from higher resolutions when needed. The hardware and system firmware must support VESA modes to allow at least the minimum desktop resolution for Windows by using the system VGA driver.

To reset a device sufficiently implies that the device is fully functional as a VGA device. It must be fully functional

when programmed via the video system firmware and when programmed directly through standard VGA registers.

Mobile Platforms only:

Mobile Multi-GPU systems must support Connecting and Configuring Displays (CCD). A Multi-GPU system may have one or more integrated and/or one or more discrete display adapters. Any display adapter which exposes an output connector to which a display device could be connected must be able to support all CCD functionality as specified in the WDDM 1.1 specification.

On systems with multiple GPUs, the laptop lid target must not be reported as "connected" on more than one graphics adapter at any given time.

Design Notes:

The following are some examples of GPU operating modes.

Linked Adapter: A linked adapter (LDA) configuration has two or more GPUs from the same vendor that are sharing the graphics workload. In a LDA configuration an application will access one GPU, but the work is distributed by the driver across multiple GPUs. Typically the application is not even aware of the existence of the multiple GPU's. Examples are NVidia SLI and AMD Crossfire.

Heterogeneous: A heterogeneous configuration consists of two or more GPUs from different GPU vendors. In a heterogeneous configuration each GPU is visible to the application.

Switch-able Graphics: A Switch-able Graphics configuration is two or more GPUs from either the same vendor or different vendors where the primary purpose is to provide multiple power/performance modes for the GPU. Examples are AMD PowerExpress, and NVIDIA Hybrid power.

The following is an example of a Hybrid switch.
When the customer unplugs the system and the GPU will switch from dGPU to iGPU, the customer has a choice to select "do not ask again for this type of GPU transition" (wording to be reviewed by Usability team later). And all future unplug will initiate a GPU switch without a dialog box.

Additional Information

Business Justification	The purpose of this requirement is to ensure core Windows usage scenarios work correctly in each GPU mode of the system. This requirement was put in place because there have been instances where the GPUs have passed certification, but once integrated into the system, the overall system would fail fundamental scenarios. One example of such a failure was a multi-GPU laptop that could not project in one of its GPU operating modes. The system had to be reconfigured manually into another operating mode by the end user. However the end user had no way of understanding or determining why the laptop was failing to project. This problem was made even worse by the fact that the system was shipped by the OEM in a mode that could not project, and the OEM software regularly placed the laptop in to the mode where the laptop could not project, even after the user had discovered the root cause.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Graphics.Display

The requirements in this section are enforced on any graphics device implementing display portion of the WDDM.

Related Requirements	• System.Fundamentals.Graphics.Display.MinimumResolutionandColorDepth

System.Fundamentals.Graphics.Display.MinimumResolutionandColorDepth

System minimum resolution and color depth

Target Feature	System.Fundamentals.Graphics.Display
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

Minimum resolution/color depth is 1024x768 at a depth of 32bits on each output simultaneously.

Additional Information

Exceptions	Tablet PC systems
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Graphics.DisplayRender

The requirements in this section are enforced on any graphics device implementing display and render portion of the WDDM.

Related Requirements	• System.Fundamentals.Graphics.DisplayRender.Performance • System.Fundamentals.Graphics.DisplayRender.StableAndFunctional

System.Fundamentals.Graphics.DisplayRender.Performance

A Windows 8 system has sufficient graphics performance to provide a good user experience for mainstream application scenarios on both A/C and battery power

Target Feature	System.Fundamentals.Graphics.DisplayRender
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64

- Windows Server 2012 x64

Description

Good graphics performance is crucial for all Windows 8 systems because users expect a smooth and responsive experience with their mainstream applications.

All Windows 8 systems must have sufficient graphics performance to meet performance requirements and metrics for a collection of scenarios that represent mainstream applications.

The tests encompass both high-level scenarios with Internet Explorer, including CSS animations, Windows Shell, and DirectX mini-scenarios.

The high-level scenarios represent end-user perceived performance for commonly used applications.

The DirectX scenarios are mini-apps with graphically rich content representing real world workloads. These mini-apps measure graphics performance using a number of common primitives that provide a proxy for real world productivity application performance.

Metrics

The following are the metrics that are measured for the applicable scenarios and used to determine the performance of a system.

Time to First Frame for each application

Time measured from process launch to first present hitting the screen. This measurement is done as a warm start.

FPS

Frames per second achieved while painting the contents of the window in a rendering loop

Memory Score

Memory score measures the total memory cost of graphics for a scenario. To calculate memory score, the driver is evaluated for memory being consumed in several scenarios. The benchmark aggregates results from different memory metrics (detailed later in this document) to arrive to a final score for the scenario.

The memory benchmark is comprised of the following six scenarios:

- Idle
- Present
- Textures
- Buffers
- Surfaces
- Upload

Glitching

For scenarios with animations, users expect smooth animations at 60 Hz. This metric looks for the total number of frames missed and number of concurrent frames missed. This is done by measuring the present events for every vsync periods between start and end of animations ignoring the first and last incomplete vsync period. The metric measurement also verifies that DWM glitches are zero.

Design Notes

These tests run on battery power only if the system is a laptop or tablet/convertible. If the system is a desktop, the tests are run on A/C only.

The following scenarios are instrumented and have goals for the metrics described:

DirectX scenario tests

Scenario test	Metrics		
	Time to first frame goals	Glitching	FPS

Blank Direct2D App **Tests the bare minimum time and resources required to just get Direct2D loaded and a putting a blank windows up.**	200 milliseconds Note: The time to create the user mode and kernel mode device by the driver should be 30 milliseconds or less for this test. The UMD and KMD creation times are part of the time to first frame measurements.	not applicable	not applicable

Direct2D Brush
Draw a bitmap with the following brush types:
• **Bitmap brush with different extend modes (clamp, wrap and mirror)**
• **Solid color brush**
• **Gradient brush**
  ○ **Linear gradient**
  ○ **Radial gradient**
• **Image brush**
	200 milliseconds for bitmap brush, solid color brush, and gradient brush; 300 milliseconds for image brush	No glitching at 60 fps	60 FPS

Direct2D Bitmaps with perspective transforms **Render a set of bitmaps with perspective transform applied.**	300 milliseconds	No glitching at 60 fps	60 FPS
Direct2D Layers **Composite images and primitives using D2D layers**	300 milliseconds	No glitching at 60 fps	60 FPS
Text **Tests include:** • **Rendering text on full screen.** • **It cycles between different rendering modes and glyph run sizes**	300 milliseconds	No glitching at 60 fps	60 FPS
Pan and Zoom **Rendering a 21 MegaPixel image and performing pan and zoom operations at the native resolution of the display.**	300 milliseconds	No glitching at 60 FPS	60 FPS
Panning through small images with effects **Test render a set of unique 200 x 200 pixel images that have image effects applied at the native resolution of the display, and pan through them in a continuous loop.**	300 milliseconds	No glitching at 60 FPS	60 FPS

Vector Graphics **Rendering animations of complex vector graphics with Direct2D.** **Note that if the underlying GPU supports feature level 11.1, then more content is rendered. Direct2D takes advantage of Target Independent Rasterization, which should provide corresponding rendering speed increases.**	300 milliseconds	No glitching at 60 FPS	60 FPS
Internet Explorer • **Load and render content representative of dynamic HTML5 content from a local on disk cache.** **Content representative of dynamic HTML5 + CSS content.**	N/A	No glitching at 60 fps	60 FPS

Bandwidth test

The GPU bandwidth is evaluated by measuring the time it takes the GPU to compose the client area of a single application with simple content (for example: no overlap, no glass) at the native resolution of the panel. The test doesn't try to achieve peak bandwidth measurement under ideal circumstances, but is meant to measure a real world scenario that includes other necessary overhead such as setting up the hardware, context switching the GPU from the application context to the DWM context and reporting completion of the rendering operation through interrupt/fence mechanism.

The goal for passing the **Bandwidth test** is dependent on the native resolution of the panel. The bandwidth is required to be sufficient to performat least 7.5 memory I/O operations per pixel at 60 FPS (or 450 * resolution.width * resolution.height * resolution.bpp). For example, this translates to the required bandwidth for the two common resolutions listed below:

Resolution	Required Bandwidth
1366x768	1.8 GB/sec
1920x1080	3.6 GB/sec

The test multiplies the number of pixels by7.5to provide for7.5 32-bit memory operations per pixel. If the measured bandwidth is greater than the required bandwidth, the test passes.

GPU bandwidth is measured with the following scenarios:

1. The DWM composes a single texture. This causes the GPU to perform the following memory operations per pixel:
    - 1 32-bit read
    - 1 32-bit write
2. The DWM composes N textures with a pixel shader which samples from N textures. This causes the GPU to perform the following memory operations per pixel:
    - N 32-bit reads
    - 1 32-bit write

    The number of textures tested (N) is the set {2, 3}.
3. The DWM composes 3 primitives in 3 separate rendering operations:

- An opaque constant color is written. This causes the GPU to perform the following memory operations per pixel:
    - 1 32-bit write
- A transparent texture is composed on top (with alpha blending enabled). This causes the GPU to perform the following memory operations per pixel:
    - 2 32-bit reads
    - 1 32-bit write
- A second transparent texture is composed on top (with alpha blending enabled). This causes the GPU to perform the following memory operations per pixel:
    - 2 32-bit reads
    - 1 32-bit write

    In aggregate, this scenario performs 4 reads and 3 writes per pixel.

4. The DWM composes an opaque solid color primitive followed by a transparent textured primitive. A 16-bit depth buffer is used, which causes the transparent primitive to be completely occluded. This causes the GPU to perform the following memory operations per pixel:
    - 2 16-bit reads
    - 1 16-bit write
    - 1 32-bit write
5. An application renders a single BC (block compressed) texture onto a full-screen surface. This causes the GPU to perform the following memory operations per pixel:
    - 1 4-bit (BC1) or 8-bit read (BC2/BC3)
    - 1 32-bit write

    To account for time spent decompressing BC textures, the goal for this scenario is adjusted to be the same as the goal for the scenario that reads from a 32-bit texture (scenario 1).
6. An application composes 2 textures with a pixel shader which samples from both textures. 1 texture uses the DXGI_FORMAT_R8_UNORM pixel format, while the other texture uses the DXGI_FORMAT_R8G8_UNORM pixel format. This causes the GPU to perform the following memory operations per pixel:
    - 1 8-bit read
    - 1 16-bit read
    - 1 32-bit write

    To account for time spent doing computation in the pixel shader, the goal for this scenario is adjusted to be the same as the goal for the scenario that reads from a single 32-bit texture (scenario 1).

Memory Benchmark Test

The memory benchmark is comprised of the following six scenarios:

- Idle
- Present
- Textures
- Buffers
- Surfaces
- Upload

For each of the rendering scenarios, the goals are established by calculating:

- 2MB allowed for OS overhead
- +2MB allowed for driver overhead per GPU in a link for x86 and x64 platforms; +1MB per GPU in a link allowed for ARM platforms. The number of linked GPUs on a system is determined and multiplied by the system-determined size per GPU to determine the total driver overhead allowed.
- +Size of surfaces explicitly created by application + 7.5% for alignment and padding in the case of non-power of two surfaces.
- +4KB overhead per surface created on 32-bit platforms; +8KB overhead per surface on 64-bit platforms.
- Except for the Present scenario, the rendering scenario targets are rounded to the next half-megabyte boundary.

Idle

The Idle scenario represents the memory cost of the driver shortly after boot on a minimal desktop with no application running other than the standard OS processes and the benchmark itself.

The score is comprised of two elements:

- Memory footprint of the driver itself in memory (non-paged code and data page)
- Outstanding resource allocation by driver (pool, contiguous memory, mdl, locked memory) excluding video memory and limited to the kernel mode driver.

The goal for the memory footprint is to be less than 10 MB and the goal for the outstanding resource allocation is to be less than 10 MB. The final benchmark idle score is less than the following:

	x86 Goal	ARM Goal	x64 Goal
Idle	Less than 20 MB	Less than 20 MB	Less than 20 MB

Note that the benchmark only takes into account the final score.

Present

The present scenario represents the base memory cost of a simple D3D device.

The test creates a 300x300 window, clears its back buffer and z-buffer, then present. When taking into account window decoration, the client area of the application is 284x261 on a default desktop.

The goal for a single GPU in this scenario is computed as follows:

- 2MB OS Overhead
- +2MB Allowed for driver / GPU overhead (x86/x64); +1MB (ARM)
- +284x261*4*1.075 DWM Redirection surface
- +284x261*4*1.075 Application Back Buffer
- +284x261*2*1.075 Application Z-Buffer (D16)
- +3 * 4096 For surface overhead (ARM/x86); +3 * 8192 (x64)
- Goal: less than 3.8MB (ARM), less than 4.8MB (x86/x64)

Textures

The textures scenario evaluates the memory overhead of creating large number of texture objects.

In addition to the resources described in the Present scenario, this scenario creates 5000 textures of 128x128 pixels using a few different pixel formats. The application renders with each texture to ensure they are referenced in the scenario.

The goal for a single GPU in this scenario is computed as follows:

- 4.8MB base cost (x86/x64); 3.8MB base cost (ARM)
- +5000*128*128*BytesPerPixel for surface memory
- +5000*4096 for surface overhead (ARM/x86); +5000*8192 (x64)

The following table lists the goals for each combination of platform and pixel format:

Format	x86 Goal (MB)	x64 Goal	ARM Goal
DXGI_FORMAT_R8G8B8A8_UNORM	336.8	356.4	335.8
DXGI_FORMAT_BC1_UNORM	63.4	82.9	62.4
DXGI_FORMAT_BC2_UNORM	102.5	122.0	101.5
DXGI_FORMAT_BC3_UNORM	102.5	122.0	101.5
DXGI_FORMAT_R8_UNORM	102.5	122.0	101.5
DXGI_FORMAT_R8G8_UNORM	180.6	200.1	179.6

Buffers

The buffers scenario represent the memory overhead of creating large number of small vertex buffers.

In addition to the resources described in the Present scenario, this scenario creates 5000 vertex buffer of 16320 bytes. The application renders with each of the vertex buffer to ensure they are referenced in the scenario.

The goal for a single GPU in this scenario is computed as follows:

- 4.8MB base cost (x86/x64); 3.8MB (ARM)
- +5000*16536 for surface memory
- +5000*4096 for surface overhead (ARM/x86); +5000*8192 (x64)
- Goal: less than 102.5MB (ARM), less than 103.5MB (x86), less than 123MB (x64)

Surfaces

The surfaces scenario represents the memory overhead a hypothetical desktop system would hit if all GDI bitmaps were converted to DX surfaces.

In addition to the resources described in the Present scenario, this scenario creates a large number of surfaces of various size and pixel depth. The file SimpleD3D10\surfaces.txt contains a snapshot (width * height * bpp) for all GDI bitmap in existence on a particular busy system. The test creates and uses surfaces of the size and pixel depth specified in this file.

The goal for a single GPU in this scenario is computed as follows:

- 4.8MB base cost (x86/x64); 3.8MB (ARM)
- +143372288*1.075 for surface memory
- +1581*4096 for surface overhead (ARM/x86); +1581*8192 (x64)
- Goal: less than 157MB (ARM), less than 158MB (x86), less than 164.5MB (x64)

See the Perl script ParseSurfaceList.pl in the memory benchmark directory to see how surface memory is computed from the PerfX\MemoryBenchmark\SimpleD3D10\Media\surface.txt file.

Upload

The upload scenario represents the memory overhead involved when uploading large amount of data to the GPU. This ensures that the driver doesn't allocate too many staging surfaces and that staging surfaces are offered as part of the upload. The test allow up to 4MB of staging to be allocated, all of it must be offered.

In addition to the resources described in the Present scenario, this scenario creates 100 textures of 512x512 of format DXGI_FORMAT_R8G8B8A8_UNORM and DEFAULT usage with some initial data. These textures are updated after first render with 512x512 data to verify that the data upload path does not incur additional driver overhead.

The goal for a single GPU in this scenario is computed as follows:

- 4.8MB base cost (x86/x64); 3.8MB (ARM)
- +1MB initial data allocated by test
- +100*512*512*4 for surface memory
- +100*4096 for surface overhead (ARM/x86); +100*8192 (x64)
- (Offered memory is automatically subtracted, up to 4MB).
- Goal: less than 105.5MB (ARM), less than 106.5MB (x86), less than 107MB (x64)

Goal Summary

The table below summarizes the goals for each of the scenario for all platforms.

Scenario	Limit ARM (MB)	Limit x86 (MB)	Limit x64 (MB)
Idle	20	20	20
Present	3.8	4.8	4.8
Textures	336	337	356.5
Buffers	102.5	103.5	123
Surfaces	157	158	164.5
Upload	105.5	106.5	107

Design Notes

If the system has any of the following configurations:

- Multiple GPUs
- It is powered via an AC adapter and used away from an outlet using a rechargeable battery for continuous operation when not plugged in

Then validation of this requirement must be done against each configuration that applies to the system.

If the system has support for connecting multiple displays, then the validation of this requirement must be done against the primary display used for the HCK system configuration.

Additional Information

Exceptions	If implemented on Server should there be a Display only or Full WDDM graphic devices in the system.
Business Justification	Graphics performance is critical to deliver a good end-user experience on Windows 8. The performance of the HW subsystems and the graphics driver plays a big role in that. The impact can be felt by the user in the following way: High memory footprint impacts overall scalability of desktop based DX usage Long Device creation time affects the application launch time Driver has much higher CPU overhead in typical usage leading to poor experience on a typical workload.
Enforcement Date	Jun. 26, 2013

System.Fundamentals.Graphics.DisplayRender.StableAndFunctional

Display device functions properly and does not generate hangs or faults under prolonged stress

Target Feature	System.Fundamentals.Graphics.DisplayRender
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

The system must run under prolonged stress without generating hangs or faults.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Graphics.HybridGraphics

Hybrid Graphics Feature

Related Requirements	• System.Fundamentals.Graphics.HybridGraphics.MultiGPU

System.Fundamentals.Graphics.HybridGraphics.MultiGPU

Hybrid Graphics

Target Feature	System.Fundamentals.Graphics.HybridGraphics
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

New systems shipping in Windows 8 that expect to be hybrid capable must adhere to the following requirements:

- A Microsoft supported hybrid system can only be composed as one integrated GPU(iGPU) and one discrete GPU(dGPU)
- Cannot have more than two GPUs
- Both GPU's must be DX11.0 or higher
- Both GPU's driver must be WDDM1.3 or higher
- Both GPUs must implement the new standard allocation type added to the KM and associated UM DDIs to support cross adapter shared surfaces.
- If each GPU has separate standard drivers, then they must be independent of each other and can be updated independently without breaking hybrid functionality
- The dGPU must be equal or higher performance than the iGPU
- The iGPU is the adapter that has outputs
- The dGPU adapter is one that

    0. sets the new discrete hybrid cap added to the WDDM 1.3 driver

    1. has no outputs

All other multi-GPU configurations do not get Microsoft hybrid support. They will be treated the same way as defined by the "System.Fundamentals.Graphics.MultipleOperatingMode" requirement.

D-list requirements

This is the list of applications maintained by the dGPU IHV for choosing a GPU for an app to run in hybrid mode or not.

- The D-list DLL can be loaded into a process and queried at most once during D3D initialization
- The DLL size must be under 200KB
- The DLL must be able to return the GPU selection choice within 4ms

Power Management Requirements

Following are the power management requirements for the discrete GPU participating in a hybrid configuration:

- The driver is required to register for runtime power management
- The driver needs to register certain power components based on the following scenarios

Device does not require D3 transitions	• **DXGK_POWER_COMPONENT_ENGINE component for each GPU engine (node)** • **A DXGK_POWER_COMPONENT_D3_TRANSITION component with one F state**
Device requires D3 transitions and has no self-refresh memory	• A DXGK_POWER_COMPONENT_D3_TRANSITION component with two F states • DXGK_POWER_COMPONENT_ENGINE component for each GPU engine (node) • A DXGK_POWER_COMPONENT_MEMORY component for each memory segment. If TransitionalLatency of this component is > 200us, component must also have DXGK_POWER_COMPONENT_FLAGS::DriverCompletesFStateTransition flag set
Device requires D3 transitions and has self-refresh memory	• A DXGK_POWER_COMPONENT_D3_TRANSITION component with two F states • DXGK_POWER_COMPONENT_ENGINE component for each GPU engine (node) • A DXGK_POWER_COMPONENT_MEMORY component for every memory segment and with the DXGK_POWER_COMPONENT_FLAGS::ActiveInD3 flag set. This component must report 2 F States and TransitionalLatency of F1 state must be 0 • One DXGK_POWER_COMPONENT_MEMORY_REFRESH component for the adapter. Also, the driver must leave space in dependency array for all device engines

- Transitional Latency reported for each component must not be greater than max. Latency tolerance for that component is specified in the table below:

	Latency tolerance
Engine (monitor ON)	
Initial state	0.08 ms
After 200 ms of idle time	15 ms
No context on the engine	30 ms
Engine (monitor OFF)	
Initial state	2 ms
After 200 ms of idle time	50 ms
No context on the engine	100 ms
Memory	
Active context exists	15 ms
No active context exists	30 ms
Memory refresh	
Initial state	0.08 ms
No active context exists	30 ms
Monitor off and no active context exists	80 ms
D3 transition	
Initial state	0.08 ms
After 10 s of all engines idle time	15 ms
No active context	200 ms
Monitor off and (no active context or all engines idle for 60 s)	250 ms

Additional Information

Enforcement Date	Jun. 26, 2013

System.Fundamentals.Graphics.InternalDisplay

Base for Graphics on Systems

Related Requirements	• System.Fundamentals.Graphics.InternalDisplay.NativeResolution

System.Fundamentals.Graphics.InternalDisplay.NativeResolution

Systems with integrated displays must use native resolution by default

Target Feature	System.Fundamentals.Graphics.InternalDisplay
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

A system with an integrated display must support the native resolution of the display and use native resolution as the default.

An "integrated" display is any display that is built into the system. A laptop lid is an example of an integrated display.

Windows is designed to work best in native resolution.
This requirement applies to systems that use UEFI or BIOS.

Additional Information

Business Justification	This requirement ensure that a Windows system can boot into the preferred native resolution without screen flashing or other artifacts
Enforcement Date	Jun. 26, 2013

System.Fundamentals.Graphics.MultipleDevice

Requirements which apply to systems with more than one graphics device.

Related Requirements	• System.Fundamentals.Graphics.MultipleDevice.Configure • System.Fundamentals.Graphics.MultipleDevice.SubsystemDeviceID

System.Fundamentals.Graphics.MultipleDevice.Configure

On a system with multiple graphics adapters, system firmware will allow the user to configure the usage of the adapters

Target Feature	System.Fundamentals.Graphics.MultipleDevice
Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

On a system with multiple graphics adapters, the system firmware (BIOS, UEFI, etc), must provide the user with the ability to modify the following settings:

- Enable/Disable any adapter
  - The firmware must offer the user the ability to select which adapter is enabled or disabled
  - At any given time at least one adapter, that supports POST, must be enabled
  - If the user enables an adapter, and the system only supports one active adapter at a time, then all other adapters must be disabled
  - If the only enabled adapter is not detected, the firmware will, fallback to the integrated adapter. If there is no integrated adapter, then fallback to the first adapter found on the first bus
- Select the adapter to be used as POST device
  - Firmware must only allow the user to select one adapter as the POST device.
  - System is allowed to POST only on an adapter that cannot be physically removed from the system.
  - At any given time at least one adapter, that supports POST, must be enabled
  - If multiple adapters that support POST are enabled, the firmware must provide the user an option to select which one will be used for POST

Additional Information

Business Justification	It is possible that the user has multiple graphics adapters connected to the system. In such a case the user might want to selectively use only a subset of the graphics adapters. Windows depends on the POST adapter for the following scenarios: Windows setup always runs on the POST adapter Pre-OS environment always runs on the POST adapter In case there is no hardware specific WDDM driver available, the Microsoft Basic Display driver will be installed on the POST adapter
Enforcement Date	Jun. 26, 2013

System.Fundamentals.Graphics.MultipleDevice.SubsystemDeviceID

Hybrid/Switchable Graphics systems that support multiple discrete graphics adapters or chipset combination must use the same Subsystem ID

Target Feature	System.Fundamentals.Graphics.MultipleDevice
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Multiple GPU Graphics configurations that support multiple discrete graphics adapters or chipset combination must use the same Subsystem ID for each device in the configuration.

Multiple GPU Graphics systems are permitted to have heterogeneous graphics solutions in certain circumstances, thus allowing different VEN IDs.

The Multiple GPU configurations which combine GPUS from different vendors must have the same SUBSYS ID to indicate the driver packages intended for a Multiple GPU system. Should the same device be used as a single device in another system, that instance of the device must use a different unique 4part PNPid.

Examples:

1. The integrated GPU and the Discrete GPU may have different VEN ID and DEV ID, but must have the same SSID. for example:

Display Devices

Card name: InField GFX
Manufacturer: OutStanding
Chip type: RUOK Family
DAC type: Integrated RAMDAC
Device Key: Enum\PCI\VEN_AAAA&DEV_EEEE&SUBSYS_9025104D&REV_A1

Display Devices

Card name: Rocking Fast GFX
Manufacturer: Awesome Chips
Chip type: 10Q Family
DAC type: Internal
Device Key: Enum\PCI\VEN_BBBB&DEV_DDDD&SUBSYS_9025104D&REV_07

2. The GPUs that is used in a Switchable machine must use a different SSID if also used in a non-switchable machine. For example:

Display Devices

Card name: InField GFX
Manufacturer: OutStanding
Chip type: RUOK Family
DAC type: Integrated RAMDAC
Device Key: Enum\PCI\VEN_AAAA&DEV_EEEE&SUBSYS_9999104D&REV_A1
Note that the OutStanding InField GFX in #1. Is the same as the one stated in #2; however, although they are the same hardware, they must have a different SSID.

Additional Information

Business Justification	This is required to identify system as being a hybrid GPU through Online Crash Analysis.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Graphics.RenderOnly

Requirements which apply to a graphics device only implementing WDDM Render DDI's

Related Requirements	• System.Fundamentals.Graphics.RenderOnly.MinimumDirectXLevel

System.Fundamentals.Graphics.RenderOnly.MinimumDirectXLevel

Render Only device on client or server system must be Direct3D 10 capable or greater.

Target Feature	System.Fundamentals.Graphics.RenderOnly
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

If a client or server system includes a render only device, the device must be Direct3D 10 capable or greater. This device can only be supported by a WDDMv1.2 Render Only Driver. Render Only devices are not allowed as the primary graphics device on client systems. All Windows 8 client systems must have a full graphics WDDM v1.2 device as the primary boot device.

Additional Information

Business Justification	Over the years there has been an increasing focus on General-purpose computing on graphics processing units (GPGPU) scenarios for doing vast amount of complex mathematical or graphical operations. Some of the common examples of this are graphics rendering for animation, database processing, financial & astronomical calculations and oil and gas exploration. The use of GPGPU has proven benefits in

	performance, power consumption and cost.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.HAL

This feature defines Hardware Abstraction Layer (HAL) requirements for systems.

Related Requirements	• System.Fundamentals.HAL.HPETRequired • System.Fundamentals.HAL.IfCSRTPresent

System.Fundamentals.HAL.HPETRequired

System provides a high-precision event timer

Target Feature	System.Fundamentals.HAL
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

This requirement was formally known as SYSFUND-0005

Systems must implement a High Precision Event Timer (HPET) that complies with the following points of the Intel Architecture Personal Computer (IA-PC) HPET Specification:

- The main counter frequency must be greater than or equal to 10 MHz and less than or equal to 500 MHz
- The main counter must monotonically increase, except on a roll-over event.
- The main counter and comparators must be at least 32 bits wide.
- The main counter must have at least three comparators.
- All of the comparators must be able to fire aperiodic, "one-shot" interrupts.
- At least one of the comparators must be able to fire periodic interrupts.
- Each comparator must be able to fire a unique and independent interrupt.
- HPET must support edge triggering interrupts.
- Timer interrupts must not be shared in LegacyIRQRouting mode.

Additional Information

Business Justification	This timer is necessary for platforms and the operating system. Existing timers cannot provide the resolution that is necessary for existing and future applications. With this

	timer, time-sensitive applications, such as multimedia applications, have the support to make high-quality applications.
Enforcement Date	Jun. 01, 2008

System.Fundamentals.HAL.IfCSRTPresent

Signed HAL extensions are required for timers and DMA controllers that are not supported in-box

Target Feature	System.Fundamentals.HAL
Applies to	• Windows 8 Client ARM (Windows RT) • Windows 8.1 Client ARM (Windows RT 8.1)

Description

For platforms that don't implement the ARM defined Generic Interval Timer (GIT), the platform should have CSRT table with resource groups that describe the timer resources. In addition, The OS image on the platform must contain Microsoft signed HAL extensions that are properly linked to the entries in the CSRT, and these HAL extensions must register the following minimum timer resources required by Windows:

- a timer (minimum resolution of one millisecond),
- a counter (Minimum resolution 1 usec),
- an always on timer (must also be registered with a resolution of at least one millisecond), and
- an always on counter (registered with a resolution of a least 1 millisecond)

If the platform includes a system (shared) DMA controller, the CSRT must include the entries to describe this controller. In addition, the OS image on the platform must contain Microsoft signed HAL extensions that are properly linked to these entries in the CSRT, and these HAL extensions must register the DMA resources required by Windows: at least one DMA Controller, and all DMA Channels for each registered DMA Controller.

Additional Information

Business Justification	The information in the tables helps Windows identify the HAL extension module(s) that need to be loaded to support the hardware implemented on the platform. The HAL extension gets information from these tables on how the core system resources are implemented and configured on this platform, to accommodate any variations among platforms.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.ImageVerification

This feature defines Hardware Abstraction Layer (HAL) requirements for systems.

Related Requirements	• System.Fundamentals.ImageVerification.ImageVerification

System.Fundamentals.ImageVerification.ImageVerification

All ARM systems must run the Image Verification Tool

Target Feature	System.Fundamentals.ImageVerification
Applies to	• Windows 8 Client ARM (Windows RT) • Windows 8.1 Client ARM (Windows RT 8.1)

Description

MANDATORY. All ARM systems must also run the Image Verification Tool.

Additional Information

Business Justification	This requirement is needed to ensure all binaries in the image are Microsoft signed.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Input

Requirements in this section apply to HID devices that are integrated in the system.

Related Requirements	• System.Fundamentals.Input.I2CDeviceUniqueHWID • System.Fundamentals.Input.PS2UniqueHWID

System.Fundamentals.Input.I2CDeviceUniqueHWID

I2C connected HID devices must have a Unique HWID along with a HID compatible ID

Target Feature	System.Fundamentals.Input
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

All I2C connected HID devices must have a unique HWID and a HID compatible ID that will allow WU to identify the device (when needed) and allow drivers to be loaded from WU.
Design Notes: See Microsoft published HID I2C protocol specification (link not provided yet)

Additional Information

Exceptions	This requirement is only enforced for HID I2C devices and not generalized for SPB.

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Input.PS2UniqueHWID

All PS/2 connected devices (such as internal keyboards) must have a unique hardware ID

Target Feature	System.Fundamentals.Input
Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64

Description

All PS/2 connected devices (such as touchpads and keyboards) must have a unique hardware ID that enables the third party driver to ship with WU.

Design Notes: See Microsoft unique hardware ID whitepaper
http://www.microsoft.com/whdc/device/input/mobileHW-IDs.mspx

Additional Information

Business Justification	This requirement is needed to enable third-party touchpad drivers to be eligible for WU. A unique hardware ID will enable the system to determine the corrected driver needed for a particular device and download it through WU
Enforcement Date	Jun. 01, 2012

System.Fundamentals.MarkerFile

A marker file" is used to help associate WER data with specific computer models. Requirements in this section describe the syntax for the "marker file.""

Related Requirements	• System.Fundamentals.MarkerFile.SystemIncludesMarkerFile

System.Fundamentals.MarkerFile.SystemIncludesMarkerFile

System includes marker file

Target Feature	System.Fundamentals.MarkerFile
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

The marker file gives additional information regarding the maker of the PC system and model. This information is used to collect and distribute On-line Crash Analysis information. The marker file is a text file with a .mrk extension. The .MRK filename must be under 256 characters in length including the path. The characters must be letters, numbers, periods, hyphens, commas and parentheses.
The marker file format is:
For companies with PCI Vendor IDs:

VendorID_CompanyName_Division_Marketing Model Name_other info.MRK

For companies without a PCI Vendor ID
CompanyName_Division_Marketing Model Name_other info.MRK

Each column is separated by the underscore '_' character. The values in each column are

VendorID = The PCI vendor ID for the PC manufacturer.
CompanyName = Name of the company go here. This should be consistent for each marker file.
Division = this represents the division within the company. If your company doesn't not have divisions please put 'na.'
Marketing Model Name = product name the system will be shipped as. This should be the same as the marketing name entered at the time of logo submission.
Other info = optional ad can be added by putting more underscores. The additional fields may be used for identifying any other critical information about the system.
Optionally, the _I field can be used as a part number that can be used to link the marketing model name to.

Design Notes:
The marker file goes in the c:\windows\system32\drivers folder.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Network

These are system level requirements that may impact the integration with a type of network device.

Related Requirements	• System.Fundamentals.Network.NetworkListOffloads • System.Fundamentals.Network.PowerRequirements

System.Fundamentals.Network.NetworkListOffloads

Wireless LAN networking device on systems that support Connected Standby must support NDIS 6.30 and support offloads

Target Feature	System.Fundamentals.Network

Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The following requirements apply to wireless LAN devices.

WLAN Devices must support the following features:

Feature	Requirement
No Pause on Suspend	Required
D0 offload	Required
USB Selective Suspend	Required- If USB based
Network List offload	Required
Wi-Fi PSM	Required
Wi-Fi Direct	Required
Radio Management	Required
WPS 2.0	Required
WoWLAN	Required

Systems that support Connected Standby require the use of an NDIS 6.30 Ethernet driver. The device must support the features listed below:

Feature	Required
Wakeup on LAN	Yes
D0 & D3 Protocol Offloads (Protocols Jun. 26, 2013)	Yes
Interrupt Moderation	Yes
OS-programmable packet filtering	Yes

Additional Information

Business Justification	Wi-Fi Network List Offload is a feature where certain Wi-Fi profile information is offloaded to the NIC firmware to allow the Wi-Fi NIC to perform logic that optimizes the power efficiency and connectivity of a given system. It helps improve the scanning and connection timings.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.Network.PowerRequirements

All physical network devices in a system (inclusive of docking stations) must meet device certification criteria for power management requirements

Target Feature	System.Fundamentals.Network

Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Support of this feature is Required.
All physical network devices included in a system (inclusive of docking stations) must meet the device-level power management requirements for that specific device type.

Example: If an Ethernet device is included in a Connected Standby capable system or associated dock, that Ethernet device must meet the power management requirements for Connected Standby regardless of whether the individual device certification was achieved when tested on a Connected Standby capable system or not.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.NX

NX is a feature that allows marking of memory pages as executable or non-Executable. This allows the CPU to help prevent execution of malicious data placed into memory by an attacker.

Related Requirements	• System.Fundamentals.NX.SystemIncludesNXProcessor

System.Fundamentals.NX.SystemIncludesNXProcessor

Systems must ship with processors that support NX and include drivers that function normally

Target Feature	System.Fundamentals.NX
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

To ensure proper device and driver behavior in systems all drivers must operate normally with Execution Protection Specifically, drivers must not execute code out of the stack, paged pool and session pool. Additionally, drivers must not fail to load when Physical Address Extension (PAE) mode is enabled, a requirement for operation of NX.

In addition, the system firmware must have NX on and data execution prevention (DEP) policy must not be set to "always off."

Additional Information

Business Justification	Microsoft Windows offers a number of defensive enhancements designed to protect customers. This technology prevents code from executing in data segments.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.PowerManagement

Power management is a feature that turns the PC off or into a lower power state. Requirements in this section describes requirements around power management.

Related Requirements	• System.Fundamentals.PowerManagement.DockUndock • System.Fundamentals.PowerManagement.MultiPhaseResume • System.Fundamentals.PowerManagement.PCResumesInTwoSeconds • System.Fundamentals.PowerManagement.PCSupportsS3S4S5 • System.Fundamentals.PowerManagement.PowerProfile

System.Fundamentals.PowerManagement.DockUndock

System supports docking and undocking across a hibernate transition

Target Feature	System.Fundamentals.PowerManagement
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

For systems which ship with a dock, the system must be able to hibernate and resume when changing from the docked to undocked state or the undocked to the docked state. This is not limited to, but should include that the memory map should not change when docking or undocking the system.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Fundamentals.PowerManagement.MultiPhaseResume

Storage subsystem supports multi-phase resume from Hibernate

Target Feature	System.Fundamentals.PowerManagement
Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64

Description

The driver and hardware subsystems for the boot storage device must support multi-phase resume from Hibernate. In order to do this, the system must be able to maintain the system's ability to identify definitively all of the memory needed on resume. This is not limited to, but should include that:

- Any crashdump filters/minifilters that must support read
- No WHEA pshed plugins are installed
- Hypervisor is not enabled

Additional Information

Exceptions	The boot storage device is booted from a USB bus.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.PowerManagement.PCResumesInTwoSeconds

System resumes from ACPI S3 state in less than specified time

Target Feature	System.Fundamentals.PowerManagement
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64

Description

The total system Standby (S3) resume time from CPU execution to resume complete, as measured by Windows® system tracing, must be two seconds or less. Total system resume time equals the sum of the system firmware S3 re-initialization time and Windows device initialization time, but does not include user mode initialization. "Resume complete" is defined as the point at which all system devices have completed their S0 power I/O Request Packet (IRP), but does not include the time required for devices to complete their D0 IRP or to become fully functional. A power test tool that includes arming and processing system trace events for resume time will be included in the WDK and will be used to test system resume time.
Exceptions:

- Systems equipped with FBDIMM memory must complete S3 resume in 7 seconds or less.
- Systems with dual symmetrical CPU sockets must complete S3 resume in 3 seconds or less.
- Systems equipped with internal USB hub (excluding USB root hub) devices must complete S3 resume in at most 2 seconds plus the time required for the USB hub devices to initialize*.

Design Notes:
See "Windows On/Off Transitions Solutions Guide" at the following website:
http://go.microsoft.com/fwlink/?LinkId=61994
The test for this requirement will automatically detect USB hub devices. USB hub devices are defined as USB-enumerated devices with Plug and Play (PnP) compatible hardware IDs that contain the string "USB\Class_09". The initialization time for each USB hub device will be calculated as the time required for the Microsoft USB hub functional device object (FDO) driver to receive and complete the S0 IRP for the USB hub device. The test will

subtract this initialization time for each USB hub device from the total system resume time. The result must be less than 2 seconds.

Additional Information

Business Justification	This requirement continues Microsoft® investment in driving fast system S3 resume performance. Fast system startup is a fundamental performance attribute required to enable key platforms and scenarios including laptop and Slate PCs, battery life, power management, media center, and small office/home office (SOHO) servers. Fast system startup is also part of the InstantOn initiative.
Enforcement Date	Jun. 01, 2006

System.Fundamentals.PowerManagement.PCSupportsS3S4S5

Systems support S3, S4 and S5 states

Target Feature	System.Fundamentals.PowerManagement
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

A desktop or mobile system installed with a client operating system must support the S3 (Sleep), S4 (Hibernate) and S5 (Soft-off) states. Every system must support wake from all implemented sleep states. Wake from S5 is only required from the power button.

A mobile system installed with a client operating system must support either S3 (Sleep) or Connected Standby. Systems that support Connected Standby must also support S4 (Hibernate).

If a USB host controller is implemented on the system, then at least one external port on the controller must support wake-up capabilities from S3. If the system contains multiple USB host controllers, all host controllers integrated on the system board (that is, not add-on cards) must support wake-up from S3. USB host controllers are not required to support wake-up when a mobile system is running on battery power.

Server systems are not required to implement S3, S4, or S5 states. If a server system does implement S3, S4 or S5 states, they must work correctly.

Power Management is an important aspect of good user experience. The system should be able to control what devices to put into a sleep state when not being used. All devices must comply with the request from the system to go into a sleep state and not veto the request thereby putting an additional drain on the power source.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Fundamentals.PowerManagement.PowerProfile

System must report form factor via power management profile

Target Feature	System.Fundamentals.PowerManagement
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The Preferred_PM_Profile in the FADT table must be set to one of the values based on the form factor of the system as outlined in the ACPI specification version 5.0. This value shall not be unspecified (0).

Design Notes:

For more information see page 119 of the ACPI specification version 5.0.

Additional Information

Business Justification	OSPM must be able to detect the form factor in order to optimize power policy defaults and tag anonymous system usage telemetry data.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.PowerManagement.CS

Power management is a feature that turns the PC off or into a lower power state. Requirements in this section describes requirements around power management for systems that support connected standby.

Related Requirements	• System.Fundamentals.PowerManagement.CS.ConnectedStandby • System.Fundamentals.PowerManagement.CS.CSQuality • System.Fundamentals.PowerManagement.CS.FanOff

System.Fundamentals.PowerManagement.CS.ConnectedStandby

System supports the connected standby power profile must set the FACP flags

Target Feature	System.Fundamentals.PowerManagement.CS
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The follow bits must be set in the FACP flags:

FACP - Field	Bit Len.	Bit Offset	Description
LOW_POWER_S0_IDLE_CAPABLE	1	21	This flag indicates if the system supports low power idle states in the ACPI S0 state. A value of one (1) indicates that the platform supports sufficiently low S0 idle power such that transitions to the S3 state are not required. OSPM may interpret a one in a manner that it favors leaving the platform in the S0 state with many devices powered off over the S3 state when the user is no longer interacting with the platform.
Reserved	10	22	*Reserved for future use.*

Additional Information

Business Justification	A separate capability for the connected standby feature is required because platforms may both expose the ACPI S3 and S4 idle states and be capable of ultra-low S0 idle power. Similarly, the platform form factor is insufficient to determine if the platform is capable of connected standby as notebook, desktops and server platforms may eventually all become systems that support connected standby.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.PowerManagement.CS.CSQuality

Systems that support Connected Standby must meet reliability standards for Runtime Power Management

Target Feature	System.Fundamentals.PowerManagement.CS
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Systems that support Connected Standby must meet minimal reliability standards as tested for this requirement. The test associated with this requirement will exercise any installed Power-Engine Plug-In (PEP), installed device drivers and platform firmware.

Design Notes

To help ensure the reliability of a system that supports connected standby, the system will be subjected to the following tests:

- Connected Standby Stress with IO
- Runtime Power Focused Stress with IO

These tests will be run while Driver Verifier is enabled with standard settings.
These tests will also be run separately with the Driver Verifier Concurrency Testing setting.
If a PEP device is enumerated in ACPI namespace and the system does not have a PEP loaded, the test will fail.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.PowerManagement.CS.FanOff

If a connected standby system has a fan, the fan must be off while the system is connected standby.

Target Feature	System.Fundamentals.PowerManagement.CS
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

All Connected Standby systems have requirements regarding thermal regardless of processor architecture and form factor. These requirements are tested for in the HCK.

1. All CS systems must have at least one thermal zone.
2. At least one thermal zone has critical shutdown temperature defined.
3. Each thermal zone must report actual temperature on the sensor.
4. All CS systems with fans must expose its activity to the OS.
5. All CS systems with fans must keep the fan off while in “Sleep” or Connected Standby state.

Active Cooling

From the point of view of the OS, a platform has two strategies that it can use to implement fan control:

1. **Implement one or more ACPI thermal zones with active trip points to engage/disengage the fan.**

   The Windows Thermal Framework supports active cooling devices at a very basic level. The only device supported inbox is an ACPI fan, and it can only be controlled with on/off signal.

2. **Implement a proprietary solution to control the fan (via drivers, an embedded controller, etc.).**

   While Windows does not control the behavior of proprietary solutions for fans, Windows does support fan notifications to thermal manager for all implementations including embedded controllers so that diagnostic information and telemetry can be collected. Thus fan exposure to the OS is required for all Connected Standby systems and strongly recommended for all others.

Note that the implementation for active cooling is completely separate from the passive cooling mitigations discussed above.

ACPI Thermal Zone Controlled Fans

Windows supports the ACPI 1.0 D-state based fan definition. Please refer to the ACPI spec for details. Thus the control is limited to fan on/off. The driver for the fan is provided in acpi.sys.

- The temperature sensor reads the temperature has crossed a trip point and issues Notify(0x80) on the associated thermal zone.
- The thermal zone reads the temperature with the _TMP control method and compares the temperature to the active trip points (_ACx) to decide whether the fan needs to be on or off.
- The OS puts the fan device in D0 or D3 which causes the fan to turn on or off.

Multiple Speed Fan in ACPI

In order to achieve multiple speeds for a fan using ACPI 1.0, there are two options:

1. The thermal zone can contain multiple “fans”, when only one physical fan exists. Having more “fans” on at one time translates to faster fan speed. Please see Section 11.7.2 in the ACPI 5.0 specification for an example of this option.

2. When the fan is on, it can decide for itself how fast to spin. Systems with embedded controllers, for example, can choose this option.

Proprietary Solution for Fans

Windows needs to be able to detect fan activity with either implementation. When a platform uses the ACPI thermal model, Windows is responsible for turning the fan on and off and therefore already knows when it is active. When a proprietary solution is used to control the fan, Windows needs notification that the fan is running. To enable this, Windows will support a partial subset of the ACPI 4.0 fan extensions, outlined in Table 6.

Feature	Description	Supported
_FST	Returns the fan’s status.	yes
Notify(0x80)	Indicates the fan’s status has changed.	yes
_FIF	Returns fan device information.	no
_FPS	Returns a list of fan performance states.	no
_FSL	Sets the fan performance state (speed).	no

Table 6: ACPI 4.0 fan object support in Windows

Windows will use the _FST object to determine if the fan is running (Control field is nonzero) or off (Control field is zero). Windows will also support Notify(0x80) on the fan device as an indication that _FST has changed and needs to be reevaluated.

A fan that implements the _FST object is not required to be in a thermal zone's _ALx device list, but it is allowed to be. This allows for a hybrid solution, where a fan is typically controlled by a third party driver, but can be controlled by the OS thermal zone if the third party driver is not installed. If a fan is in an _ALx device list and is engaged by the thermal zone (placed in D0), the _FST object is required to indicate a non-zero Control value.

For all fans, Windows will determine the state of the fan using the following algorithm:

1. If a fan is in D0 (as a result of a thermal zone's _ACx trip point being crossed), it is engaged.
2. If a fan is in D3 and does not support the ACPI 4.0 extensions, it is disengaged.
3. If a fan is in D3 and supports the ACPI 4.0 extensions, the OS will check _FST's Control field for a non-zero value to see if the fan is engaged

Fan Presence

A platform indicates that there is a fan on the system by including a fan device (PnP ID PNP0C0B) in the ACPI namespace. Windows will take the presence of this device as an indication that the system has a fan, and the absence of this device as an indication that the system has no fan.

Connected Standby Specific Guidance

The Windows Thermal Management Framework is a part of the kernel and ships with all Windows systems. Thus, the material above applies to all machines. However, various types of machines require additional guidance more specific to Connected Standby.

Connected Standby brings the smartphone power model to the PC. It provides an instant on, instant off user experience that users have come to expect on their phone. And just like on the phone, Connected Standby enables the system to stay fresh, up-to-date, and reachable whenever a suitable network is available. Windows 8 supports Connected Standby on low-power platforms that meet specific Windows Certification requirements.

Connected-Standby capable systems are highly mobile devices in a thin and light form factor. Further, Connected Standby systems are always on and in the ACPI S0 state. In order to deliver a robust and reliable customer experience, the entire system —from the mechanical design to firmware and software implementation— must be designed with critical attention to thermal characteristics.

The system must include a temperature sensor for the SoC at a minimum.

Processor Throttling

The PPM communicates the maximum, desired and minimal performance levels to the PEP. Under thermal throttling conditions, the maximum performance level should equal the throttling performance requested by the thermal manager. The PEP then sets the CPU's physical voltage and frequency based on the PPM's performance level requirements.

Design Notes

For more information, please see the Thermal Guide document.

Additional Information

Exceptions	This requirement only applies to connected standby systems.
Business Justification	From a user perspective, the machine appears "off". When the system is in connected standby, users expect it to be equivalent to the "sleep" action. Thus the fan is expected to never come on, just as in traditional machines during sleep. If the fan does come on, users may hear it and/or feel the hot air circulating and think that the computer has a bug. Therefore, the fan must not come on while running a realistic

	Connected Standby workload in a standard lab environment.
Enforcement Date	Jan. 01, 2014

System.Fundamentals.PXE

The Preboot Execution Environment (PXE) is an environment to boot computers using a network interface.

Related Requirements	• System.Fundamentals.PXE.PXEBoot

System.Fundamentals.PXE.PXEBoot

Remote boot support for PXE complies with BIOS Boot Specification 1.01 or EFI boot manager

Target Feature	System.Fundamentals.PXE
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

All systems are required to be PXE capable. The system must support booting from the network as defined in BIOS Boot Specification, Version 1.01, Appendix C, or as controlled by the EFI boot manager. This requirement is exempt for systems that are configured with Wireless LAN only.

Systems shipping with a UEFI compatible operating system and supporting PXE boot must support IPV4 PXE and IPV6 PXE booting as defined in UEFI 2.3.1.

UNDI must support :

- a DUID-UUID per IEFT draft
  - (http://tools.ietf.org/html/draft-narten-dhc-duid-uuid-01)
- DHCP6, DUID-UUID, IPv6
  IPV4 multicast

Design Notes:
Microsoft recommends that the implementation of accessing PXE be consistent with BIOS Boot Specification, Version 1.01, and Appendix C.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Fundamentals.Reliability

These are for reliability tests content from the former DEVFUND category.

Related Requirements	• System.Fundamentals.Reliability.SystemReliability

System.Fundamentals.Reliability.SystemReliability

Drivers in a system must be reliable

Target Feature	System.Fundamentals.Reliability
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

All drivers in a system must pass all requirements under Device.DevFund.Reliabilty. All systems will need to pass Common Scenario stress, sleep stress with IO and Enable/Disable with IO.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.Security

Additional TDI filter driver and LSP requirements related to security.

Related Requirements	• System.Fundamentals.Security.DeviceEncryption • System.Fundamentals.Security.NoTDIFilterAndLSP

System.Fundamentals.Security.DeviceEncryption

Systems that support connected standby must support device encryption

Target Feature	System.Fundamentals.Security
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Systems that support connected standby must meet the security requirements to support enablement of Device Encryption. OEMs must not block the enablement of Device Encryption when deploying the OS images unless the device is pre-provisioned with a third-party disk encryption solution. Device Encryption will be enabled on these systems to ensure that user data is protected. As pre-requisites for Device Encryption, connected standby systems must meet requirements for TPM and Secure Boot as outlined in System.Fundamentals.TPM20 and System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Fundamentals.Security.NoTDIFilterAndLSP

No TDI filters or LSPs are installed by the driver or associated software packages during installation or usage

Target Feature	System.Fundamentals.Security
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

There can be no use of TDI filters or LSPs by either kernel mode software or drivers, or user mode software or drivers.

Additional Information

Business Justification	Use of TDI filters and LSPs increase attack surface, and will therefore no longer be supported for future OS releases.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SignedDrivers

This feature checks for signed drivers.

Related Requirements	• System.Fundamentals.SignedDrivers.BootDriverEmbeddedSignature • System.Fundamentals.SignedDrivers.DigitalSignature

System.Fundamentals.SignedDrivers.BootDriverEmbeddedSignature

Boot drivers must be self-signed with an embedded signature

Target Feature	System.Fundamentals.SignedDrivers
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

All boot start drivers must be embedded-signed using a Software Publisher Certificate (SPC) from a commercial certificate authority. The SPC must be valid for kernel modules. Drivers must be embedded-signed through self-signing before the driver submission.

Design Notes:
For more information about how to embedded-sign a boot start driver, see "Step 6: Release-Sign a Driver Image File by Using an Embedded Signature" at the following website:

http://www.microsoft.com/whdc/winlogo/drvsign/kmcs_walkthrough.mspx

After the file is embedded-signed, use SignTool to verify the signature. Check the results to verify that the root of the SPC's certificate chain for kernel policy is "Microsoft Code Verification Root." The following command line verifies the signature on the toaster.sys file:

```
Signtool verify /kp /v amd64\toaster.sys

Verifying: toaster.sys
SHA1 hash of file: 2C830C20CF15FCF0AC0A4A04337736987C8ACBE3
Signing Certificate Chain:
Issued to: Microsoft Code Verification Root
Issued by: Microsoft Code Verification Root
Expires: 11/1/2025 5:54:03 AM
SHA1 hash: 8FBE4D070EF8AB1BCCAF2A9D5CCAE7282A2C66B3
...
Successfully verified: toaster.sys
Number of files successfully Verified: 1
Number of warnings: 0
Number of errors: 0
```

In the Windows Hardware Certification Kit, this requirement will be tested by using the Embedded Signature Verification test.

Additional Information

Business Justification	Boot drivers must be embedded-signed in order to work properly with the boot process.

Enforcement Date	Jun. 01, 2007

System.Fundamentals.SignedDrivers.DigitalSignature

System must contain logo qualified devices

Target Feature	System.Fundamentals.SignedDrivers
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

All buses, devices and other components in a system must meet their respective Windows Hardware Certification Requirements and use drivers that either are included with the Windows operating system installation media or are digitally signed by Microsoft through the Windows Hardware Certification Program that match the Windows OS version being submitted for and shipping with.

For example, if a logo qualifying a system for Windows 7, then all drivers on the system must be signed by Microsoft for Windows 7 or be drivers that ship on the Windows 7 media. All devices in the system would also need to be logo qualified or certified for Windows 7. This requirement applies to all versions of Microsoft Windows.

All devices and drivers need to be fully installed, and does not contain any problem codes.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Fundamentals.SMBIOS

System Management BIOS (SMBIOS) requirements defines data structures in the system firmware which allows a user or application to store and retrieve information about the computer.

Related Requirements	• System.Fundamentals.SMBIOS.SMBIOSSpecification

System.Fundamentals.SMBIOS.SMBIOSSpecification

System firmware support for SMBIOS complies with the SMBIOS specification

Target Feature	System.Fundamentals.SMBIOS

Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

The system firmware must implement support for SMBIOS that complies with System Management BIOS Reference Specification, Version 2.4 or later. The SMBIOS implementation must follow all conventions and include all required structures and fields as indicated in the SMBIOS Specification, Section 3.2, and follow all conformance requirements as indicated in Section 4. Bit 2 in the BIOS Characteristics Extension Byte 2 field must be set (Section 3.3.1.2.2 of the specification). The length of the Type 1 (System Information) table must be at least 1Bh bytes (includes SKU Number and Family fields from Version 2.4 of the specification).

Additionally, the following fields must have non-Null values that accurately describe the computer system or computer system component:

- (Table 0, offset 04h) BIOS Vendor
- (Table 0, offset 08h) BIOS Release Date
- (Table 0, offset 14h) BIOS Major Release Version[1]
- (Table 0, offset 15h) BIOS Minor Release Version[1]
- (Table 1, offset 04h) System Manufacturer[2]
- (Table 1, offset 05h) System Product Name[2]
- (Table 1, offset 08h) Universal Unique ID number
- (Table 1, offset 19h) System SKU Number[2]

Microsoft recommends that the following fields have non-Null values that accurately describe the computer system or computer system component:

- (Table 0, offset 05h) BIOS Version
- (Table 0, offset 16h)Embedded Controller Major Release Version[3]
- (Table 0, offset 17h) Embedded Controller Minor Release Version[3]
- (Table 1, offset 06h) System Version
- (Table 1, offset 1Bh) System Family[2]
- (Table 2, offset 04h) Base Board Manufacturer
- (Table 2, offset 05h) Base Board Product
- (Table 2, offset 06h) Base Board Version

[1]These fields must not have values equal to 0FFh.
[2]These fields gain prominence as fields which will be used for identifying unique system configurations for telemetry and servicing. The Manufacturer, Product Name, SKU Number and Family fields must not be longer than 64 characters in length. Avoid leading or trailing spaces or other invisible characters.
[3]If the system has a field upgradeable embedded controller firmware; these values should not be equal to 0FFh.

Design Notes: SKU Number has been moved to a required field in order to improve telemetry reporting. We encourage the OEM to be careful to fill in Manufacturer consistently and to fill in SKU Number with a value that can identify what the OEM considers a unique system configuration for telemetry and servicing.

Additional Information

Business Justification	It is important to do everything possible to ensure that Windows removes privacy-related information such as machine GUID, Serial Number and Asset Tag before adding SMBIOS data to crash dumps.
Enforcement Date	Jun. 01, 2006

System.Fundamentals.StorageAndBoot

This section summarizes the requirements for storage and boot devices.

Related Requirements	• System.Fundamentals.StorageAndBoot.BootPerformance • System.Fundamentals.StorageAndBoot.EncryptedDrive • System.Fundamentals.StorageAndBoot.SATABootStorage

System.Fundamentals.StorageAndBoot.BootPerformance

Boot Devices in systems that support Connected Standby must meet these requirements

Target Feature	System.Fundamentals.StorageAndBoot
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

The following requirements apply to Boot Devices in systems that support Connected Standby.

In addition to SATA and USB, Windows platform supports SD and eMMC based storage. An eMMC device may be used as either a system (boot) disk or as a data disk but an SD device can only be used as a data disk.

Flash Type	Boot	Data Disk
eMMC 4.5+	Yes	Yes
SD 2.0 or 3.0	No	Yes

ACPI interfaces must specify whether the storage device is internal or external and whether or not it is removable or fixed.

_RMV must be defined in ACPI namespace for any embedded devices attached to an SD host controller, where 0 is defined as non-removable. _RMV may optionally be defined for external slots as 1.

The following parameters must be defined within the "Storage Class-Specific Information" to be returned by the ACPI _DSM method for an SD/eMMC Storage Controller:

- Number of Sockets
- Socket Addresses

Support GPIO card detection on SD/eMMC Storage Controller.

When using eMMC as the primary boot device, the eMMC memory must be hardware partitioned such that the boot critical portion of the EFI Firmware resides in an area of the device that is not accessible by Windows.

The CPU Vendor and/or Firmware Provider must furnish the software tools needed to maintain and update the firmware.

The following requirements are applicable to boot storage media and are tested with the smaller of 2% or 1GB free space:

Feature	Span Size	Specification
Power		
Max Idle Power	-	<= 5 mW
Random Performance		
4KB Write IOPs	1 GB	>= 200
	*5 GB	>= 50
	†10 GB	>= 50
64KB Write IOPs	1 GB	>= 25
4KB Read IOPs	*5 GB	>= 2000
	†10 GB	>= 2000
4KB 2:1 read/write mix IOPs	1 GB	>= 500
	* 5GB	>= 140
	†10 GB	>= 140
Sequential Performance		
Write speed (64KB I/Os)	*5 GB	>= 40 MB/s
	†10 GB	>= 40 MB/s
Write speed (1MB I/Os)	*5 GB	>= 40 MB/s
	†10 GB	>= 40 MB/s
Read speed (64KB I/Os)	*5 GB	>= 60 MB/s
		‡>= (120 MB/s)
	†10GB	>= 60 MB/s

		‡>= (120 MB/s)
Device I/O Latency		
Max Latency	-	< 500 milliseconds

*Applies only to devices with 16 GB of internal storage or lower.

†Applies only to devices with greater than 16 GB of internal storage.

‡Applies only if the device is HS200 capable.

Additional I/O Latency requirement:

Maximum of **20 seconds** sum-total of user-perceivable I/O latencies over any **1 hour** period of a user-representative workload, where a user-perceivable I/O is defined as having a latency of at least 100 milliseconds.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Fundamentals.StorageAndBoot.EncryptedDrive

Systems which ship with a Encrypted Drive as a boot storage device must support security command protocols in order to make sure the data at rest is always protected.

Target Feature	System.Fundamentals.StorageAndBoot
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

The following requirements apply if Encrypted Drive (System.Fundamentals.StorageAndBoot.EncryptedDrive) support is implemented for a UEFI based client system or server systems:

1. The system MUST have a TPM 1.2 or TPM 2.0.
2. The system MUST implement the EFI_STORAGE_SECURITY_COMMAND_PROTOCOL from either:
    a. Trusted Command Support in UEFI 2.3 + UEFI Mantis change number 616 or
    b. UEFI 2.3.1
3. The implementation of the Trusted Computing Group Platform Reset Attack Mitigation Specification (http://www.trustedcomputinggroup.org/resources/pc_client_work_group_platform_reset_attack_mitigation_specification_version_10) MUST unconditionally issue TPer Reset (OPAL v2.0 in section 3.2.3) for all scenarios whenever memory is cleared.

4. The EFI_STORAGE_SECURITY_COMMAND_PROTOCOL and the TPer Reset command MUST be included in the base UEFI image (not in a separate image of a UEFI driver).
5. The system MUST enumerate all Encrypted Drives and TPer Reset MUST be issued prior to executing any firmware code not provided by the platform manufacturer in the base UEFI image.
6. The TPer Reset MUST be issued regardless of whether the TPM has had ownership taken or not.

Note: The TPer Reset action will occur later in the boot process than the memory clear action because it has a dependency on the EFI_STORAGE_SECURITY_COMMAND_PROTOCOL.

Additional Information

Business Justification	Support for BitLocker with Encrypted Drives (when Encrypted Drives are used with BitLocker, booting the system to the Windows Logon prompt automatically unlocks bands through BitLocker authentication process. To help ensure BitLocker'd data at rest is protected, the bands must be locked if the system is power cycled)
Enforcement Date	Mar. 01, 2012

System.Fundamentals.StorageAndBoot.SATABootStorage

System with SATA boot storage must meet requirements

Target Feature	System.Fundamentals.StorageAndBoot
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

AHCI Host Controllers using Windows for boot support must be compliant with the AHCI specification.

Host Controller Interface	Revision
AHCI	1.1, 1.2, 1.3

System with SATA controller must enable AHCI mode support.
Externally connected SATA devices (eSATA) are not supported for boot storage.

When SATA is used as the primary boot device, to ensure reliability and prevent inadvertent erasure of the firmware that may cause the device to become inoperable, the boot critical portion of the UEFI firmware must reside on a separate storage device that is not accessible by the host Operating System. The CPU Vendor and/or Firmware Provider must furnish the software tools needed to maintain and update the firmware.

When used in systems that support connected standby, the SATA device must meet the power requirements stated in the section for System.Fundamentals.StorageAndBoot.BootPerformance.

Additional Information

Enforcement Date	Aug. 01, 2012

System.Fundamentals.SystemAudio

This section contains all of the audio requirements for PCs

Related Requirements	• System.Fundamentals.SystemAudio.Audio • System.Fundamentals.SystemAudio.HardwareVolumeControl • System.Fundamentals.SystemAudio.MicrophoneLocation • System.Fundamentals.SystemAudio.NoiseOnTheSignal • System.Fundamentals.SystemAudio.SystemEmploysAntiPop • System.Fundamentals.SystemAudio.SystemMicArray • System.Fundamentals.SystemAudio.SystemUsesHDAudioPinConfigs

System.Fundamentals.SystemAudio.Audio

Systems contain audio devices that conform to Windows Logo requirements

Target Feature	System.Fundamentals.SystemAudio
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Systems need to conform to all audio device requirements

Additional Information

Exceptions	If audio is implemented in the PC, then this requirement must be met.
Enforcement Date	Jun. 01, 2006

System.Fundamentals.SystemAudio.HardwareVolumeControl

System with user exposed hardware audio volume control(s) uses approved control method to report status

Target Feature	System.Fundamentals.SystemAudio
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64

	• Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

This requirement was formerly known as SYSFUND-0057.

Non-slate/convertible systems that implement an integrated hardware volume control must use one of the following volume controls methods:

- Volume knob widget as defined in the Intel High Definition (HD) Audio Specification
- PS/2 scan codes as defined in Keyboard Scan Code Specification
- HID controls as defined in HID Audio Controls and Windows

For Slate and Convertible PCs a slider or knob is not acceptable. The volume control button must either be a rocker or two button combination with one button raising the volume level while the other lowers it.

Design Notes:
Specifically for HD Audio implementations (see the referred specifications for the other two solutions); the HD Audio specification outlines how a codec can send an unsolicited response to software whenever a volume button is pressed or whenever a rotary quadrature encoder provides a pulse to the volume knob widget pins on the HD Audio codec. It is also possible to use an ADC and get the volume position data from an analog volume pot control, but the rotary quadrature encoder pulse method is recommended. The HD Audio specification specifies two methods of using the volume knob widget: direct and indirect. Microsoft recommends using the indirect method to ensure the best user experience with the Windows operating system.

Additional Information

Exceptions	Audio volume buttons are required for Slate PCs and convertibles. If other systems and keyboards include audio volume controls then they must meet this requirement.
Business Justification	Consistent volume experience for our users is important to succeed in the field where consumer electronics parity is paramount. It is important that the Audio Engine and Automatic echo cancellation feature knows exactly what is going on between the audio device and the speakers. If there are any unknown (analog) changes to the audio signal after it leaves the control of the OS this will cause user confusion and lead to different UX on different PC systems.
Enforcement Date	Jun. 01, 2006

System.Fundamentals.SystemAudio.MicrophoneLocation

Microphone Location Reporting Requirement

Target Feature	System.Fundamentals.SystemAudio
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

1. All active onboard fixed-position single element microphones and onboard fixed-position microphone arrays (multiple combined elements) on a system must be simultaneously available for independent capture.
2. Systems with no onboard fixed-position microphone arrays, but with multiple onboard fixed-position microphones (e.g. Front/Back), AND no combined microphone, must have exactly one with GeoLocation = "Front" (which will be set as the default microphone on that system).
3. Systems with multiple onboard fixed-position microphone arrays must have exactly one with GeoLocation = "Front."
4. Mic arrays with "n" elements must deliver RAW audio to the system. The RAW format must have "n" channels, and data must come directly from the mic elements, free of signal processing.
5. If a microphone and a camera are physically co-located onboard then stated location information for each must match.
6. For devices with multiple onboard fixed-position microphones or multiple arrays, the names of these endpoints should be unique on the system. To specify a unique name, there are a few different methods using KSPROPERTY_PIN_NAME, IPinName and .inf pin description name GUID registration. Here are links that show how to implement these methods:
    a. KSPROPERTY_PIN_NAME
    http://msdn.microsoft.com/en-us/library/windows/hardware/ff565203(v=vs.85).aspx/
    b. IPinName
    http://msdn.microsoft.com/en-us/library/windows/hardware/ff536837(v=vs.85).aspx
    c. Friendly Names for Endpoints
    http://msdn.microsoft.com/en-us/library/windows/hardware/ff536394(v=vs.85).aspx

Additional Information

Exceptions	This requirement is for mobile and all-in-one form factors. Desktop machines are exempt.
Business Justification	Microphones may be found in various physical locations on any given device. In order to insure that systems and apps are able to programmatically understand where these mics are located and select the proper mic for a particular scenario, these requirements are needed.
Enforcement Date	Jun. 26, 2013

System.Fundamentals.SystemAudio.NoiseOnTheSignal

Noise on the signal from the audio device generated by system components is -80dB FS or better

Target Feature	System.Fundamentals.SystemAudio
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

This requirement was formerly known as sysfund-0048.

When measuring the audio signal on any line level audio output from the system, the audio noise level created by all of the components in the system must be minus 80dB FS regardless of PC system activity. This includes audio passed to replicate line level audio outputs in docking stations or port replicators. This requirement does not apply to audible acoustic noise from a system measured in air next to a PC system but rather to the analog noise generated on the audio signals generated by the system's streaming audio device by system components during various levels of system activity.

Design Notes: During various system loads such as high CPU usage, hard-disk activity, and CD or DVD playback, or mouse activity, network activity must not interfere with audio fidelity. The types of noise that typically create problems are mostly caused by ground loops; however, electromagnetic interference is also possible. The metric can be based either on the dynamic range or frequency response.

Additional Information

Business Justification	Improving PC audio fidelity is important to support the media usage scenarios.
Enforcement Date	Jun. 01, 2006

System.Fundamentals.SystemAudio.SystemEmploysAntiPop

System employs anti-pop measures on all system audio outputs

Target Feature	System.Fundamentals.SystemAudio
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

This requirement was formerly known as SYSFUND-0050

System employs one or more of the below anti-pop measures during power-up and power-down and all supported power state changes such as hibernate and stand-by; · Power supply is bipolar. · system firmware mute audio outputs during power-up sequence. · System employs tuned ramp-up of analog voltage supply circuit and VRef filter capacitor. · The various audio outputs on a system are off by default, by using switch or relay techniques and turned on only after an appropriate delay after power is applied to the system audio device (2-7s). This delay avoids the audible and sometimes very destructive pop/click caused by unipolar power supplies in today's PC. This requirement does not apply to primary chassis internal speaker, commonly used for PC Beep notifications.

Design Notes:
To ensure the outputs are off during power state changes, external anti-pop circuitry can be

Additional Information

Business Justification	Pops and clicks have been a problem in the PC space forever and is basically a direct result of the unipolar and otherwise inadequate power supplies used in PCs. Attempting to mask the pops with switches or relays external to the audio device is a band aid solution but short of changing the power supplies of all PCs is the best we can hope to do. Pops and clicks during power state transitions can severely damage highly sensitive and expensive audio speaker equipment and with the PC moving into the living room in larger numbers we want to ensure the PC behaves as nicely as other CE equipment when turned on or put into the various sleep modes.
Enforcement Date	Jun. 26, 2013

System.Fundamentals.SystemAudio.SystemMicArray

If the system includes microphone array hardware then it exposes the array to the Microsoft class drivers through defined methods

Target Feature	System.Fundamentals.SystemAudio
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

This requirement was formerly known as SYSFUND-0067.

Systems that include microphone array hardware must implement these according to the requirements as described in the "Microsoft Microphone Array Design" white paper. The array characteristics must be exposed to the Microsoft UAA class drivers through methods defined in the design guidelines and/or specification for each supported audio technology.
If the microphone array is implemented through USB, see the Microsoft UAA USB Audio Design Guidelines and the microphone array white paper outlining how to expose the array geometry in the descriptors of the USB audio device.
Depending on specific array geometry, report array characteristics by using the Windows audio device property set designed to expose the array geometry. See the Mic Array white paper for this info.

Design Notes:
The Microphone Array software support in Windows requires the array to meet certain design guidelines as outlined in the Microsoft Mic Array design guidelines. The guidelines have exact metrics for the array geometries but there is tolerance built into the windows mic array support.

The mic array needs to report its geometry to windows accurately.

Mic Array Whitepaper: http://www.microsoft.com/whdc/device/audio/micarrays.mspx

Audio Device Technologies for Windows http://www.microsoft.com/whdc/device/audio/default.mspx

Additional Information

Exceptions	If a Microphone array is implemented then it must meet this requirement.
Enforcement Date	Jun. 01, 2006

System.Fundamentals.SystemAudio.SystemUsesHDAudioPinConfigs

System uses the HD Audio device pin configuration registers to expose logical devices supported by the Windows UAA HD Audio class driver

Target Feature	System.Fundamentals.SystemAudio
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Enabling the Microsoft UAA HD Audio class driver to create a number of HD Audio logical audio devices is important to ensure a great standard experience when third-party drivers are not available. This is done by adhering to the Microsoft HD Audio pin configuration programming guidelines and exposing the following devices divided into areas based on audio device hardware functionality resources.
Resource scenario #1: HD Audio device with hardware resources that support one independent render and one independent capture stream. The pin configurations must expose the following devices:

- 1 independent output (speaker, headphone, redirected headphone or line out)
- 1 independent input (microphone in, line in, mixed/muxed microphone in, or mixed/muxed line in)

Resource scenario #2: HD Audio device with hardware resources that support two independent render and two independent capture streams. The pin configurations must expose the following devices:

- 2 independent outputs (speaker, headphone, redirected headphone or line out)
- 2 independent inputs or 1 independent input and 1 mixed/muxed input (microphone in or line in)

Resource scenario #3: HD Audio device with hardware resources that support four or more independent render and two or more independent capture streams (SYSFUND-0047 compliant). The pin configurations must expose the following devices:

- 1 multi-channel independent output (speaker, redirected headphone or line out)
- 1 independent output (speaker, headphone or line out)
- 2 independent inputs or 1 independent input and 1 or more mixed/muxed inputs (microphone in or line in)

Resource scenario #4: HD Audio device with hardware resources that supports digital output (SPDIF or HDMI) output and/or digital (SPDIF or HDMI) input in addition to either of the first three resource scenarios (SYSFUND-0047 compliant). The pin configurations must expose the following devices in addition to the device required by the corresponding resource scenario:

- 1 independent digital output such as SPDIF or HDMI
- 1 independent digital input (if solution has this capability) such as SPDIF or HDMI

For front panel audio jacks on a desktop or the side/front panel jacks on a mobile platform, at least one of the jacks must be defined as a Headphone output in the HD Audio codec's pin configuration register by the system firmware and function as such as the default behavior.

Design Notes: Independent resource is defined as a stream that can pass through the device without affecting any other exposed device in the system/codec.
See the Pin Configuration Guidelines for High Definition Audio Devices white paper at http://go.microsoft.com/fwlink/?LinkId=58572.

Additional Information

Business Justification	Without properly configured pin register defaults the Microsoft UAA HD Audio class driver will not be able to deliver a base level audio experience in the absence of a 3rd party driver. The justification for the class driver: System Integrators, OEMs/ODMs; Independent Hardware Vendors can depend on Microsoft provided drivers to cover Operating system upgrades & installs, older or unsupported hardware, value products, SKUs where stability and security are imperative. For end users, there is guaranteed audio support on upgrade or clean Windows installation. It provides option for stable, secure audio when advanced features are not required.
Enforcement Date	Jun. 01, 2006

System.Fundamentals.SystemAudio.3rdPartyDriver

The requirements and tests in this section are related to audio devices in a system that use a third party driver.

Related Requirements	• System.Fundamentals.SystemAudio.3rdPartyDriver.UAA

System.Fundamentals.SystemAudio.3rdPartyDriver.UAA

Audio device is compliant with one of the appropriate technology specifications supported by the UAA initiative

Target Feature	System.Fundamentals.SystemAudio.3rdPartyDriver
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64

	• Windows Server 2012 x64

Description

Audio devices in systems must comply with Device.Audio.3rdPartyDriver.UAA.

Additional Information

Enforcement Date	Sep. 17, 2008

System.Fundamentals.SystemPCIController

These requirements describe the requirements for a PCI or PCI Express controller in a system

Related Requirements	• System.Fundamentals.SystemPCIController.PCIRequirements • System.Fundamentals.SystemPCIController.SystemImplementingRiserCard

System.Fundamentals.SystemPCIController.PCIRequirements

System devices and firmware meet PCI requirements

Target Feature	System.Fundamentals.SystemPCIController
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

All PCI Express devices must comply with the PCI Base Express Specification 1.1 unless specified otherwise below.

Reverse bridge implementations as defined in Appendix A of the PCI Express to PCI/PCI-X Bridge Specification are not supported in Windows

A reverse bridge will not be supported if it adheres to the guidelines and recommendations as defined in Appendix A of the PCI Express to PCI/PCI-X Bridge Specification, Revision 1.0.

System firmware disables the extended (non-VC0) virtual channels in PCI Express devices

The system firmware sets the VC enable bit (PCI Express Base Specification, Revision 1.0a, Section 7.11.7, "VC Resource Control Register: bit 31") to 0 for all extended (non-VC0) virtual channels in all PCI Express devices. This requirement does not apply to PCI Express High Definition Audio controllers, which use class code 04 and subclass code 03.Because extended support for VC hardware is optional, this requirement addresses the scenario in which incompatible VC hardware implementations might cause system reliability, stability, and performance issues.

Hardware vendors are encouraged to work with Microsoft to define the future direction of extended virtual channel support.

System firmware for PCI-X Mode 2 capable and PCI Express systems implements MCFG table for configuration space access

PCI-X Mode 2-capable and PCI Express systems must implement the MCFG ACPI table in PCI Firmware Specification, Revision 3.0.The configuration space of PCI-X Mode 2 and PCI Express devices must be accessible through the memory-mapped configuration space region defined in this table.

PCI-to-PCI bridges comply with PCI-to-PCI Bridge Architecture Specification

All PCI-to-PCI bridges must comply with PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Virtual bridges that comply with PCI Express also comply with PCI-to-PCI Bridge Architecture Specification

PCI Express Base Specification, Revision 1.0a, virtual bridges must comply with PCI-to-PCI Bridge Architecture Specification, Revision 1.1.

In addition, VGA 16-bit decode (Section 3.2.5.18, "Bridge Control Register, bit 4") and SSID and SSVID (Section 3.2.5.13) from PCI-to-PCI Bridge Architecture Specification, Revision 1.2, must also be supported.

SSID and SSVID support is not required until January 1, 2011. If implemented, SSID and SSVID must meet the specification.

SSVID is not required for PCIe to PCI/PCI-X bridges.

Mobile system can assign distinct IRQs for at least six PCI devices

Mobile systems must be able to assign at least six distinct interrupts for devices with a PCI configuration header. Note that if the system supports MSI, any devices that implement MSI can be counted against this minimum.

x64-based system provides 64-bit support in PCI subsystem

For x64-based systems, all PCI bridges on the system board must support dual-access cycle (DAC) for inbound access, and DAC-capable devices must not be connected below non-DAC-capable bridges, such as on adapter cards.

All new 64-bit adapters must be DAC capable.

This DAC requirement does not apply to outbound accesses to PCI devices. However, for systems in which DAC is not supported on outbound accesses to PCI devices, the system firmware must not claim that the bus aperture can be placed above the 4-GB boundary.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Fundamentals.SystemPCIController.SystemImplementingRiserCard

System board implementing a riser card provides a unique ID for the riser

Target Feature	System.Fundamentals.SystemPCIController
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

The system firmware for a system board that supports any type of enumerable riser card, such as AMR, ACR, and CNR, must include the following support:

- Detecting and enumerating each function on that type of riser device.
- Representing each function on that device so the relevant Windows bus enumerator (such as a PCI bus enumerator) can detect it and then locate and install appropriate drivers.

For any riser card, the system firmware must provide a unique PCI SID assigned by the codec manufacturer. This requirement is identical to current requirements for audio and modem devices on a PCI add-on card. However, these riser cards are system-board devices, so the PCI SID must reflect that of the system-board manufacturer. If an OEM chooses a riser card and driver from any riser card driver manufacturer, the system firmware must populate the fields as follows:

- The PCI SVID must reflect the VID that the PCI-SIG assigned to that OEM.
- The SID must be unique for each AC '97 device configuration. For example, for a MoM, MR, or AMR device, each SID must be unique.

If an OEM chooses a system board from a manufacturer that works with one or more codecs, the following applies:

- The SVID must reflect the VID that the PCI-SIG assigned to that system-board manufacturer.
- The SID must be unique for each AC '97 codec and device configuration. For example, for a MoM, MR, or AMR device, each SID must be unique.
- For an AMR device, the system firmware must properly implement the detection algorithm from Intel to verify that the hardware on an AMR or MR extension is actually present.

Similar provisions exist in the CNR and ACR specifications.

Design Notes:
See the Intel documents for AC '97 at http://go.microsoft.com/fwlink/?LinkId=50733.

Additional Information

Business Justification	Required by the PCI-sig and we rely on this to load the right drivers. Drivers overall, including Windows Update, would not work without this.
Enforcement Date	Jun. 01, 2006

System.Fundamentals.SystemUSB

This section contains requirements for systems with USB host controllers.

Related Requirements	• System.Fundamentals.SystemUSB.EHCIToXHCIControllerTransitions • System.Fundamentals.SystemUSB.ExternalUSBonCSisEHCIorXHCI • System.Fundamentals.SystemUSB.SuperSpeedCapableConnectorRequirements • System.Fundamentals.SystemUSB.SuperSpeedPortsAreVisualDifferent • System.Fundamentals.SystemUSB.SuperSpeedTerminationRemainsOn • System.Fundamentals.SystemUSB.SystemExposesUSBPort • System.Fundamentals.SystemUSB.TestedUsingMicrosoftUsbStack

- System.Fundamentals.SystemUSB.USB3andUSB2PortsRoutedToSameXHCIController
- System.Fundamentals.SystemUSB.USBDevicesandHostControllersWorkAfterPowerCycle
- System.Fundamentals.SystemUSB.XhciBiosHandoffFollowsSpec
- System.Fundamentals.SystemUSB.xHCICompatibleUnlessForApprovedTargetDesigns
- System.Fundamentals.SystemUSB.XHCIControllerSaveState
- System.Fundamentals.SystemUSB.XHCIControllersMustHaveEmbeddedInfo
- System.Fundamentals.SystemUSB.xHCIControllerSupportMSIInterrupts
- System.Fundamentals.SystemUSB.XhciSupportsMinimum31Streams
- System.Fundamentals.SystemUSB.XhciSupportsRuntimePowerManagement
- System.Fundamentals.SystemUSB.XHCIToEHCIControllerTransitions

System.Fundamentals.SystemUSB.EHCIToXHCIControllerTransitions

System firmware handles hand off of transition from xHCI USB controller to eHCI USB controller

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

If implemented, system firmware may support switching a port between an EHCI or XHCI controller. However only 1 controller may be active and identified to the OS for each USB port, and this controller may be switched only at boot time. Unless the XHCI controller is explicitly disabled in BIOS, the system firmware or a software filter driver must detect if the OS supports XHCI during boot, and if the OS supports XHCI, it must re-route the USB ports to the XHCI controller.

Design Note:

To test this requirement, the user will need to plug in both a USB 3.0 and a USB 2.0 device into each USB 3.0 port that supports routing to the EHCI controller. Confirm that Windows 8 detects these devices as connected to the XHCI controller.

Additional Information

Business Justification	If the system ships with the USB compatibility mode for legacy OS enabled,this compatibility mode must be disabled for Windows 8, so that users will realize the benefits of the XHCI controller.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.ExternalUSBonCSisEHCIorXHCI

External USB ports on system that support connected standby must be EHCI or XHCI

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

USB host controllers on systems that support Connected Standby must implement xHCI (eXtensible Host Controller Interface) or EHCI (Enhanced Host Controller Interface). Legacy companion controllers (UHCI/OHCI) are not supported.

All exposed ports must support all speeds slower than the maximum speed of the host controller, to enable support of legacy devices including keyboards and mice.

Required Speed Support	EHCI Port (USB 2.0)	XHCI Port (USB 3.0)
Low-Speed	Yes	Yes
Full-Speed	Yes	Yes
Hi-Speed	Yes	Yes
Super-Speed	No	Yes

Transaction translators (TTs), integrated with the EHCI host controller, are not standardized, but the Windows EHCI driver supports several implementations of a controller- integrated TT. The supported integrated TT implementation must be identified in ACPI using the _HRV hardware revision for the USB controller. Please contact the USB team to determine if your implementation is supported and for more information about which _HRV value should be reported.

If the USB EHCI controller does not feature an integrated TT, any externally exposed ports must be routed through an embedded rate-matching hub.

For improved power efficiency and performance, USB Host Controllers on systems that support Connected Standby are recommended to be at least USB 3.0 compatible, with an XHCI controller integrated into the SoC or chipset. The operating system supports standard EHCI and XHCI controllers including debug registers.

USB Host Controller Interface	Recommendation
UHCI/OHCI Companion Controllers	Not-supported
EHCI	Supported
XHCI (including debug capability)	Supported and Recommended

Additional Information

Business Justification	For improved power efficiency and performance, Windows will not support legacy UHCI and OHCI controllers on systems that support Connected Standby. This requirement also ensures that USB keyboard and Mice will work as expected when connected to the USB port. USB 3.0 ports connected to a standard XHCI controller automatically satisfy this requirement.
Enforcement Date	Jun. 01, 2012

System.Fundamentals.SystemUSB.SuperSpeedCapableConnectorRequirements

Each exposed SuperSpeed capable connector supports SuperSpeed, high, full and low speed USB devices routed through its xHCI controller

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

xHCI Controllers are backwards compatible with SuperSpeed, high, full, and low speed USB devices. Backwards compatible is defined as all USB devices enumerate and function at their intended speeds. More than one xHCI controller may be present on a system as long as the SuperSpeed capable ports are correctly routed. EHCI controllers may also be present on the system, however SuperSpeed capable ports should not be routed through them.

Additional Information

Business Justification	This requirement ensures that low, full, and high speed devices continue to work as xHCI controllers become more prevalent in systems.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.SuperSpeedPortsAreVisualDifferent

Systems with SuperSpeed Ports and non-SuperSpeed ports must have visual differentiation between the two types of ports

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Version update: In the second paragraph changed a should to must.

If the system under test has exposed mixed ports (both SuperSpeed and non-SuperSpeed ports, such as USB 2.0 ports, exposed to the user)the SuperSpeed ports must be visually distinguishable from the non-SuperSpeed ports.

Non-SuperSpeed ports must not be marked blue as blue is reserved in the USB 3.0 specification (Section 5.3.1.3) as an optional marking for SuperSpeed ports.

Additional Information

Business Justification	The intent of this requirement is to allow users to be able to clearly distinguish their SuperSpeed ports on a mixed port system to get optimal performance from their SuperSpeed devices. Note that we are only asking for this differentiation if both types of ports are exposed to the user.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.SuperSpeedTerminationRemainsOn

SuperSpeed termination remains on once power is applied to the bus, unless the OS explicitly removes it.

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

Once the system firmware applies power to the bus, for all xHCI ports, the SuperSpeed termination remains on unless the OS explicitly removes it.

Design note:

To test this requirement, the user will need to plug in a USB 3.0 peripheral device (that is, a non-hub). The device must meet the following requirement. Given these circumstances:

- Device is connected to a USB 3.0 connector
- Device is operating at USB 2.0
- The connector's SuperSpeed termination transitions from Disabled to Enabled, but there is NOT a USB 2.0 reset

The test may fail if the peripheral device connects over USB 3.0 under these circumstances, because such a device cannot be used to validate this requirement. The device must wait for a USB 2.0 reset before attempting to connect over USB 3.0. For more information about the expected behavior of peripheral devices, see section 10.16.1 of the Universal Serial Bus 3.0 Specification.

Additional Information

Business Justification	If SuperSpeed terminations are removed, the device enumerates over USB 2.0 and there is a delay for the device to be re-enumerated over USB 3.0.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.SystemExposesUSBPort

Systems are recommended to expose at least one user-accessible USB port

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Systems are recommended to expose at least one external USB host port. If a system exposes such port, the following requirement applies:

For the best experience with Windows, systems that expose such port are recommended to have an externally accessible standard USB A port. This allows users to connect their existing USB devices, without an adapter. Standard USB A ports can also be used as a kernel debug port, to expose standard USB EHCI Host debug, USB XHCI host debug, or USB function debugging.

While USB 2.0 capable controllers are acceptable, USB 3.0 XHCI host controllers are preferred. The USB ports must fully comply with the USB 3.0 or USB 2.0 specification. USB 3.0 connectors must properly support 900mA USB 3.0 devices, and 500 mA USB 2.0 and 1.1 devices. USB 2.0 ports must properly support 500 mA USB 2.0 and 1.1 devices.

Windows supports several dual-role (OTG) USB controllers with an EHCI compliant host controller implementation. These dual-role controllers can be used primarily in host mode with the inbox Windows USB controller drivers. When properly configured, several of these supported dual-role controllers can be used as a kernel debug transport in function mode.

If the system's form factor is too thin to expose a standard USB A port, it is acceptable to expose a micro-A/B port. See the table below for the complete list of options.

It is optional to include adapter that converts the port from micro USB to USB A. If you bundle an adapter, the adapter speed must match that of the USB host controller. For example, if the host controller is a USB 3.0 xHCI controller, then the adapter must support USB 3.0. This adapter enables a user to connect a standard USB device to the micro-USB A/B port or proprietary docking port and port and must ground the USB ID pin of the port.

External USB Ports	Recommendation
Standard USB A Port(s)	Recommended
Micro-USB A/B (Host + Function debug) Port	Supported
Micro-USB B Port + 1 or more Standard USB A Port	Supported
Proprietary Docking Port with USB Host and/or debug Functionality	Supported

Mini-USB A, A/B or B Port	Not Supported
Proprietary USB Host Port	Not Supported
Micro-USB A/B Port + 1 or more Standard USB A Port	Not Supported

Whatever USB port type is chosen, it must be correctly described in ACPI with the _UPC (USB Port Capabilities) package type parameter as defined in the ACPI 4.0a specification, section 9.13. This information allows Windows to determine when a micro-A/B port is exposed, and ID pin detection is necessary.

A simple Standard USB A male to Micro USB B female adapter can be used to expose USB function or XHCI host debug transport from a Standard USB-A port. This adapter must prevent shorts on the VBus line by removing the VBus line completely or by having a 1kOhm resistor inline with the VBus line. It is strongly recommended that the standard USB A port provide built-in protection against a short on the VBus line. This can occur if the USB port is connected to another host when it is not properly configured in debug mode.

If there is a standard USB A (host) port in addition to a micro-USB B (function debug) port, the USB ports must be connected to separate USB controllers. Thus, the micro-USB B (function) port can be connected to a USB OTG controller in function mode while the standard USB A (host) port would be connected to a USB host controller.

If the micro-USB B port provides no additional functionality beyond debugging, it must be hidden in the battery compartment or behind a easily removable cover. In order to comply with USB-IF requirements, VBUS must not be asserted on the micro-A/B port until the resistance to ground of the ID pin of the micro-USB A/B port is less than 10 Ohms. This will prevent a short-circuit when a user connects a micro-USB B cable to another USB host, such as a desktop. Alternatively, the port can implement short protection circuitry for VBus.

Additional Information

Business Justification	Allowing users to use their existing USB devices is a differentiating feature of Windows. The micro-USB port or proprietary docking port allow thinner form factors to be designed.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.TestedUsingMicrosoftUsbStack

Systems with xHCI Controllers must be tested with Microsoft's xHCI Stack installed

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Systems with Extensible Host Controller Interface (xHCI) Controllers must be tested with Microsoft's xHCI Stack installed and enabled.

Additional Information

Business Justification	All USB host controllers must be compatible with the Microsoft inbox driver stack.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.USB3andUSB2PortsRoutedToSameXHCIController

Systems which have xHCI controllers, should route the USB 3.0 (Super Speed) and 2.0 port corresponding to each connector to the same xHCI Controller

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

For systems that have xHCI controllers, any 2.0 and 3.0 (SuperSpeed) ports that are share a connection point must be routed to the same xHCI controller. This requirement applies to all USB connection points, whether they are externally visible or not.

This would explicitly prevent "companion controller" implementation of xHCI controllers in systems.
Companion controller implementation for xHCI should be avoided because it can cause device issues such as bug checks, bad device states as well as devices going missing as well as provide a bad user experience. xHCI drivers should be able to handle all device speeds and types and thus a companion controller should not be necessary when integrating an xHCI controller into a system.
Please see design notes for additional information.

Design Notes:
Referenced in the xHCI v1.x specification, section 4.24.2.2

Additional Information

Business Justification	This reduces the risk of a system crash due to fast switch of devices between USB ports.
Enforcement Date	Dec. 01, 2010

System.Fundamentals.SystemUSB.USBDevicesandHostControllersWorkAfterPowerCycle

All USB devices and host controllers work properly upon resume from sleep, hibernation or restart without a forced reset of the USB host controller

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT)

	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

All USB devices and host controllers work properly upon resume from sleep, hibernation or restart without a forced reset of the USB host controller

Design Notes: Registry key ForceHCResetOnResume documented at the KB below is not needed for devices to function properly upon resume in Windows 7 or newer.
http://support.microsoft.com/kb/928631
Note that a known set of currently existing devices do require a forced reset upon resume, these devices should be covered in a list kept by the OS which will reset these devices upon resume. The goal of this requirement is to ensure that this list of devices which need a reset to appear after resume does not grow and that devices can properly handle sleep state transitions without being reset.
A reset of the entire USB Host Controller results in significantly increased time that it takes for all USB devices to become available after system resume since there could be only one device at address 0 at a time, this enumeration has to be serialized for all USB devices on the bus. We have also seen that resetting the host controller can lead to an illegal SE1 signal state on some host controllers, which in turn can cause some USB devices to hang or drop off the bus. Moreover, devices cannot maintain any private state across sleep resume as that state will be lost on reset.

Additional Information

Business Justification	This registry value was created as a fix for devices that were not coming back upon resume.
Enforcement Date	Dec. 01, 2010

System.Fundamentals.SystemUSB.XhciBiosHandoffFollowsSpec

xHCI BIOS handoff follows specification

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

For all xHCI controllers exposed to the OS, the system firmware must follow the BIOS handoff procedure defined in section 4.2.2.1 of the XHCI specification.

Additional Information

Business Justification	Driver software expects the BIOS to perform handoff compliant to specification.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.xHCICompatibleUnlessForApprovedTargetDesigns

System that support Connected Standby must implement SuperSpeed ports as defined in the xHCI and SuperSpeed specification or use approved EHCI reference design

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Systems that support Connected Standby that have USB connectors must enable USB 3 capabilities as defined in the SuperSpeed and xHCI specification or they must use approved EHCI reference design for USB 2 capabilities.

Additional Information

Business Justification	USB ports on the system enhance the end user experience so they can use their peripherals with the system and having an XHCI USB host controller would be able to take full advantage of the latest USB 3.0 devices.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.XHCIControllerSaveState

xHCI controllers correctly save and restore state, or else indicates an error

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Refer to sections 4.23.2.1, 5.4.1, and 5.4.2 of the xHCI specification version 1.0, plus any applicable errata. Upon completion of the Controller Restore State operation, if the controller has not successfully restored its internal state to the appropriate state, then it must set the Save/Restore Error bit to '1'. Errors could be due to power loss during low system power states, or other conditions.

Testability/how to test: Simplest way to test (uses usbxhci): Connect a device to xHCI, suspend/resume the system, look for ETW errors such as "Slot not enabled" in the resume path.

To get better coverage (uses compliance stack):

1. Have no devices enumerated

2. Read number of device slots
3. Send an Enable Slot command for each device slot
4. Attempt one extra Enable Slot command€”it should return "No slots available"
5. Save state
6. Put the system in a low power state and wake it
7. Restore state
8. Attempt one extra Enable Slot command€”it should return "No slots available"
9. Disable all previously-enabled slots (should succeed)

Additional Information

Business Justification	Software is tolerant of this error if it is reported by the hardware in accordance with the spec. If the error is not reported, further unexpected errors will occur later, because the host controller driver will continue to have an incorrect view of the state of the host controller. No performant mitigations exist that cover all ways in which a controller could fail silently; therefore we will test for some failures in the HCK, and publish this requirement to highlight the importance of the error mechanism for save/restore state.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.XHCIControllersMustHaveEmbeddedInfo

Systems with xHCI controllers must have embedded ACPI information for port routing

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Please reference ACPI specification version 4.0.

The ACPI namespace hierarchy for USB devices should exactly match the devices hierarchy enumerated by Windows operating system.

All connectable USB ports are required to have a _PLD object. In addition, two fields in the _PLD object: Group token (Bit 86:79) and Group Position (Bit 94:87) should be correctly defined for all USB connection points, including those that are not visible externally (which should be indicated by setting bit 64 to zero).

No two USB connection points should have identical combination of Group token and Group Position. If two ports are sharing a connection point, they should have identical _PLD objects.

This information helps define the mapping of USB ports to uniquely identifiable connection points. The Windows USB 3.0 stack will use this information to determine which ports are tied to the same connection points. Any USB port that does not have a _PLD object will be assumed to be not connectable and not visible (i.e. it is not being used at all). The definition of connectable port as per ACPI 4.0 spec (section 9.13), is a port on which either a user can connect a device OR there is an integrated device connected to it.

Please see design notes for additional information on how to implement this requirement.

Design Notes:
Example
This example is based on xHCI Spec (Version .95) Appendix D. The hardware configuration is exactly the same as in that Appendix. The ACPI representation of that hardware configuration differs in this example; those differences are highlighted.
The following is an example of the ACPI objects defined for an xHCI that implements a High-speed and SuperSpeed Bus Instance that are associated with USB2 and USB3 Protocol Root Hub Ports, respectively. The xHCI also supports an integrated High-speed hub to provide Low- and Full-speed functionality. The External Ports defined by the xHC implementation provide either a USB2 data bus (i.e. a D+/D- signal pair) or a SuperSpeed (or future USB speed) data bus (i.e. SSRx+/SSRx- and SSTx+/SSTx-signal pairs).
Where:

- The motherboard presents 5 user visible connectors C1 - C5.
    - Motherboard connectors C1 and C2 support USB2 (LS/FS/HS) devices.
    - Motherboard connectors C3, C4 and C5 support USB3 (LS/FS/HS/SS) devices.
- The xHCI implements a High-speed Bus Instance associated with USB2 Protocol Root Hub ports, e.g. HCP1 and HCP2 are High-speed only, i.e. they provide no Low- or Full-speed support.
- The xHCI presents 7 External Ports (P1 - P7).
    - External Port 1 (P1) is HS only and is not visible or connectable.
    - External Ports 2 - 5 (P2 - P5) support LS/FS/HS devices.
        - P2 is attached to motherboard USB2 connector C1.
        - P3 is attached to motherboard USB2 connector C2.
        - P4 is attached to the USB 2.0 logical hub of the Embedded USB3 Hub on the motherboard. The USB 2.0 logical hub supports the LS/FS/HS connections for 2 ports (EP1 - EP2)
        - The USB 2.0 connections of motherboard hub ports EP1 and EP2 are attached to motherboard connectors C3 and C4 respectively, providing the LS/FS/HS support for the USB3 connectors.
    - P5 is attached to motherboard connector C5, providing the LS/FS/HS support to the motherboard USB3 connector C5.
    - External Port 6 (P6) is attached to the SuperSpeed logical hub of the Embedded USB3 Hub on the motherboard. The SuperSpeed logical hub supports the SS connections of 2 ports (EP1 - EP2).
    - The SuperSpeed connections of motherboard hub ports EP1 and EP2 are attached to motherboard connectors C3 and C4 respectively, providing the SS support for the USB3 connectors.
    - External Port 7 (P7) is attached to motherboard connectors C5, providing the SS support for the USB3 connector.
- The xHCI implements 4 internal HS Root Hub ports (HCP1 - HCP4), 2 High-speed and 2 SuperSpeed.
    - Internal Port 1 (HCP1) maps directly to External Port 1 (P1).

- Internal Port 2 (HCP2) is attached to a HS Integrated Hub. The Integrated Hub supports 4 ports (IP1 - IP4).
    - Ports 1 to 4 (IP1-IP4) of the Integrated Hub attach to External Ports 2 to 5 (P2-P5), respectively.
- Internal Ports 3 and 4 (HCP3, HCP4) attach to External Ports 6 and 7 (P6, P7), respectively.

- All connectors are located on the back panel and assigned to the same Group.
- Connectors C1 and C2 are USB2 compatible and their color is not specified. Connectors C3 to C5 are USB3 compatible and their color is specified.
- External Ports P1 - P5 present a USB2 data bus (i.e. a D+/D- signal pair). External Ports P6 and P7 present a SuperSpeed data bus (i.e. SSRx+/SSRx- and SSTx+/SSTx- signal pairs).

```
Scope( \_SB ) {

        Device( PCI0 ) {

// Host controller ( xHCI )
Device( USB0 ) {
// PCI device#/Function# for this HC. Encoded as specified in the ACPI
// specification
Name( _ADR, 0xyyyyzzzz )
// Root hub device for this HC #1.
Device( RHUB ) {
Name( _ADR, 0x00000000 ) // must be zero for USB root hub
// Root Hub port 1 ( HCP1 )
Device( HCP1 ) {// USB0.RHUB.HCP1
Name( _ADR, 0x00000001 )
// USB port configuration object. This object returns the system
// specific USB port configuration information for port number 1
Name( _UPC, Package() {
0x01,// Port is connectable but not visible
0xFF,// Connector type (N/A for non-visible ports)
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
} // Device( HCP1 )
// Root Hub port 2 ( HCP2 )
Device( HCP2 ) {// USB0.RHUB.HCP2
Name( _ADR, 0x00000002 )
Name( _UPC, Package() {
0xFF,// Port is connectable
0x00,// Connector type - (N/A for non-visible ports)
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// Even an internal connection point should have a _PLD and
// provide a valid Group Token and Position
Name( _PLD, Buffer( 0x10) {
0x00000081,// Revision 1,
// color width height ignored for non-visible connector
0x00000000,// connector type ignored for non-visible connector
```

```
0x00808000,// Not User visible, Panel, position shape ignored,
// Group Token = 1, Group Position = 1
// This is the group of all internal connectors.
// Each connector should have a unique position in this
// group
0x00000000} )// Ignored for non-visible connectors
//
// There is no separate node for the integrated hub itself
//
// Integrated hub port 1 ( IP1 )
Device( IP1 ) {// USB0.RHUB.HCP2.IP1
// Address object for the port. Because the port is
// implemented on integrated hub port #1, this value must be 1
Name( _ADR, 0x00000001 )
Name( _UPC, Package() {
0xFF,// Port is connectable
0x00,// Connector type - Type 'A'
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// provide physical connector location info
Name( _PLD, Buffer( 0x10) {
0x00000081,// Revision 1, Ignore color
// Color (ignored), width and height not
0x00000000,// required as this is a standard USB 'A' type
// connector
0x00800c69,// User visible, Back panel, Center, left,
// shape = vert. rect, Group Token = 0,
// Group Position 1 (i.e. Connector C1)
0x00000003} )// ejectable, requires OPSM eject assistance
} // Device( IP1 )
// Integrated Hub port 2 ( IP2 )
Device( IP2 ) {// USB0.RHUB.HCP2.IP2
// Address object for the port. Because the port is
// implemented on integrated hub port #2, this value must be 2
Name( _ADR, 0x00000002 )
Name( _UPC, Package() {
0xFF,// Port is connectable
0x00,// Connector type - Type 'A'
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// provide physical connector location info
Name( _PLD, Buffer( 0x10) {
0x00000081,// Revision 1, Ignore color
// Color (ignored), width and height not
0x00000000,// required as this is a standard USB 'A' type
// connector
0x01000c69,// User visible, Back panel, Center, Left,
// Shape = vert. rect, Group Token = 0,
// Group Position 2 (i.e. Connector C2)
0x00000003} )// ejectable, requires OPSM eject assistance
} // Device( IP2 )
// Integrated Hub port 3 ( IP3 )
Device( IP3 ) { // USB0.RHUB.HCP2.IP3
// Address object for the port. Because the port is implemented
```

```
// on integrated hub port #3, this value must be 3
Name( _ADR, 0x00000003 )
// Must match the _UPC declaration for USB0.RHUB.HCP3 as
// this port shares the connection point
Name( _UPC, Package() {
0xFF,// Port is not connectable
0x00,// Connector type - (N/A for non-visible ports)
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// Even an internal connection point should have a _PLD and
// provide a valid Group Token and Position.
// Must match the _PLD declaration for USB0.RHUB.HCP3 as
// this port shares the connection point
Name( _PLD, Buffer( 0x10) {
0x00000081,// Revision 1,
// color width height ignored for non-visible connector
0x00000000,// connector type ignored for non-visible connector
0x01008000,// Not User visible, Panel, position shape ignored,
// Group Token = 1, Group Position = 2
// This is the group of all internal connectors.
// Each connector should have a unique position in this
// group
0x00000000} )// Ignored for non-visible connectors
//
// There is no separate node for the embedded hub itself
//

// Motherboard Embedded Hub 2.0 Logical Hub port 1 ( EP1 )
Device( EP1 ) {// USB0.RHUB.HCP2.IP3.EP1
Name( _ADR, 0x00000001 )
// Must match the _UPC declaration for
// USB0.RHUB.HCP3.EP1 as this port provides
// the LS/FS/HS connection for C3
Name( _UPC, Package() {
0xFF,// Port is connectable
0x03,// Connector type - USB 3 Type 'A'
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// provide physical connector location info
Name( _PLD, Buffer( 0x10) {
0x0072C601,// Revision 1, Color valid
// Color (0072C6h), width and height not
0x00000000,// required as this is a standard USB
// 'A' type connector
0x01800c69,// User visible, Back panel, Center,
// Left, shape = vert.
// rect, Group Token = 0,
// Group Position 3
//(i.e. Connector C3)
0x00000003} )// ejectable, requires OPSM eject
// assistance
} // Device(EP1)
// Motherboard Embedded Hub 2.0 Logical Hub port 2 ( EP2 )
Device( EP2 ) {// USB0.RHUB.HCP2.IP3.EP2
```

```
Name( _ADR, 0x00000002 )
// Must match the _UPC declaration for
// USB0.RHUB.HCP3.EHUB.EP2 as this port provides
// the LS/FS/HS connection for C4
Name( _UPC, Package() {
0xFF,// Port is connectable
0x03,// Connector type - USB 3 Type 'A'
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// provide physical connector location info
Name( _PLD, Buffer( 0x10) {
0x0072C601,// Revision 1, Color valid
// Color (0072C6h), width and height not
0x00000000,// required as this is a standard USB
// 'A' type connector
0x02000c69,// User visible, Back panel, Center,
// Left, Shape = vert.
// rect, Group Token = 0,
// Group Position 4 (i.e. Connector C4)
0x00000003} )// ejectable, requires OPSM eject
//assistance
} // Device( EP2 )
} // Device( IP3 )

// Integrated hub port 4 ( IP4 )
Device( IP4 ) { // USB0.RHUB.HCP2.IP4
Name(_ADR, 0x00000004)
// Must match the _UPC declaration for USB0.RHUB.HCP4 as
// this port provides the LS/FS/HS connection for C5
Name( _UPC, Package() {
0xFF,// Port is connectable
0x03,// Connector type - USB 3 Type 'A'
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// provide physical connector location info
Name( _PLD, Buffer(0x10) {
0x0072C601,// Revision 1, Color valid
// Color (0072C6h), width and height not
0x00000000,// required as this is a standard USB 'A' type
// connector
0x02800c69,// User visible, Back panel, Center, Left,
// Shape = vert. rectangle, Group Token = 0,
// Group Position 5 (i.e. Connector C5)
0x00000003} )// ejectable, requires OPSM eject assistance
} // Device( IP4 )
} // Device( HCP2 )

// Root Hub port 3 ( HCP3 )
Device( HCP3 ) {
Name( _ADR, 0x00000003 )
// Must match the _UPC declaration for USB0.RHUB.HCP2.IP3 as
// this port shares the connection point
Name( _UPC, Package() {
0xFF,// Port is connectable
```

```
0x00,// Connector type - (N/A for non-visible ports)
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// Even an internal connection point should have a _PLD and
// provide a valid Group Token and Position.
// Must match the _PLD declaration for USB0.RHUB.HCP2.IP3 as
// this port shares the connection point
Name( _PLD, Buffer( 0x10) {
0x00000081,// Revision 1,
// color width height ignored for non-visible connector
0x00000000,// connector type ignored for non-visible connector
0x01008000,// Not User visible, Panel, position shape ignored,
// Group Token = 1, Group Position = 2
// This is the group of all internal connectors.
// Each connector should have a unique position in this
// group
0x00000000} )// Ignored for non-visible connectors
//
// There is no separate node for the embedded hub itself
//
// Motherboard Embedded Hub SS Logical Hub port 1 ( EP1 )
Device( EP1 ) {// USB0.RHUB.HCP3.EP1
Name( _ADR, 0x00000001 )
// Must match the _UPC declaration for
// USB0.RHUB.HCP2.IHUB.IP3.EHUB.EP1 as this port
// provides the SS connection for C3
Name( _UPC, Package() {
0xFF,// Port is connectable
0x03,// Connector type - USB 3 Type 'A'
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// provide physical connector location info
Name( _PLD, Buffer( 0x10) {
0x0072C601,// Revision 1, Color valid
// Color (0072C6h), width and height not
0x00000000,// required as this is a standard USB
// 'A' type connector
0x01800c69,// User visible, Back panel, Center,
// Left, shape = vert.
// rect, Group Token = 0,
// Group Position 3
//(i.e. Connector C3)
0x00000003} )// ejectable, requires OPSM eject
// assistance
} // Device(EP1)
// Motherboard Embedded Hub SS Logical Hub port 2 ( EP2 )
Device( EP2 ) {// USB0.RHUB.HCP2.EP2
Name( _ADR, 0x00000002 )
// Must match the _UPC declaration for
// USB0.RHUB.HCP3.IP3.EP2 as this port
// provides the SS connection for C4
Name( _UPC, Package() {
0xFF,// Port is connectable
0x03,// Connector type - USB 3 Type 'A'
```

```
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// provide physical connector location info
Name( _PLD, Buffer( 0x10) {
0x0072C601,// Revision 1, Color valid
// Color (0072C6h), width and height not
0x00000000,// required as this is a standard USB
// 'A' type connector
0x02000c69,// User visible, Back panel, Center,
// Left, Shape = vert.
// rect, Group Token = 0,
// Group Position 4 (i.e. Connector C4)
0x00000003} )// ejectable, requires OPSM eject
// assistance
} // Device( EP2 )
} // Device( HCP3 )
// Root Hub port 4 ( HCP4 )
Device( HCP4 ) {// USB0.RHUB.HCP4
Name( _ADR, 0x00000004 )
// Must match the _UPC declaration for USB0.RHUB.HCP2.IP4 as
// this port provides the SS connection for C5
Name( _UPC, Package() {
0xFF,// Port is connectable
0x03,// Connector type - USB 3 Type 'A'
0x00000000,// Reserved 0 - must be zero
0x00000000} )// Reserved 1 - must be zero
// provide physical connector location info
Name( _PLD, Buffer( 0x10) {
0x0072C601,// Revision 1, Color valid
// Color (0072C6h), width and height not
0x00000000,// required as this is a standard USB 'A' type
// connector
0x02800c69,// User visible, Back panel, Center, Left,
// Shape = vert. rect, Group Token = 0,
// Group Position 5 (i.e. Connector C5)
0x00000003} )// ejectable, requires OPSM eject assistance
} // Device( HCP4 )
} // Device( RHUB )

} // Device( USB0 )
//
// Define other control methods, etc.

} // Device( PCIO )

} // Scope( \_SB )
```

Additional Information

Business Justification	We need this info to be correct so we know which of the available ports are SuperSpeed capable. We'll use this to direct the user to the correct port when they connect a SuperSpeed device.

Enforcement Date	Dec. 01, 2010

System.Fundamentals.SystemUSB.xHCIControllerSupportMSIInterrupts

xHCI Controllers support MSI and/or MSI-X interrupts

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

USB xHCI host controllers (eXtensible Host Controller Interface) in systems must support and use MSI and/or MSI-X interrupts as defined in section 6.8 of the PCI Local Bus Specification Revision 3.0 and Section 5.2.6 of the xHCI specification.

Additional Information

Business Justification	This is to ensure compliance with the industry specification
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.XhciSupportsMinimum31Streams

xHCI controller must support at least 31 primary streams per endpoint

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

Refer to the eXtensible Host Controller Interface specification, section 4.12.2.

This requirement is for the MaxPSASize in the HCCPARAMS to be set to 4 at the minimum to enable ultimate data transfer rate with UAS devices.

Storage devices based on the USB Attached SCSI Protocol (UASP) will utilize streams to achieve faster data transfer rates. To enable the best experience with these devices, every xHCI controller will need to support at least 31 primary streams.

Additional Information

Business Justification	USB based storage devices based must be USB Attached SCSI Protocol (UASP) and will utilize streams to achieve faster data transfer rates.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.XhciSupportsRuntimePowerManagement

All USB xHCI host controllers support runtime power management including, if implemented, runtime wake

Target Feature	System.Fundamentals.SystemUSB
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

All USB xHCI host controllers in the system must support runtime power management, as required by the eXtensible Host Controller Interface specification, version 1.0, Section 4.15.

Runtime is defined as the system working state (S0), including the Connected Standby sub-state of S0 if Connected Standby is supported.

Power management of the host controller encompasses software-initiated idle power down (controller low power state such as D3), software-initiated power up, and, optionally, hardware-initiated wake signaling.

For each xHCI controller that is reported to support runtime wake signaling, the controller must be able to wake itself successfully upon any of the following events:
A) Any suspended port detecting device wake signaling, or
B) Any port detecting connect, disconnect, or overcurrent, when the corresponding PORTSC Wake on Xxx bit is set to '1'.
For more details, see Section 4.15 of the xHCI specification.

The system plays a role in delivering xHCI wake signals properly. Therefore, the system must correctly report, via ACPI, whether each xHCI controller is capable of waking at runtime. This information must be reported in the ACPI _S0W object.

Additional Information

Business Justification	USB Power Management is critical to battery life.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.SystemUSB.XHCIToEHCIControllerTransitions

Once the power is applied to the bus, the SuperSpeed termination remains on unless the OS explicitly removes it

Target Feature	System.Fundamentals.SystemUSB

Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

Once the power is applied to the bus, the SuperSpeed termination remains on unless the OS explicitly removes it.

System firmware supporting EHCI<->xHCI mode switch enter xHCI mode properly and are always in xHCI mode when Windows 8 boots.

"Properly" is defined as:
1) select xHCI
2) transition USB bus from VBUS off to VBUS enabled
3) leave connectors' SuperSpeed terminations enabled for the entire time between steps 2 and 4
4) handoff xHCI to the OS.

Design note:

To test this requirement, the user will need to plug in a USB 3.0 peripheral device (that is, a non-hub). The device must meet the following requirement. Given these circumstances:

- Device is connected to a USB 3.0 connector
- Device is operating at USB 2.0
- The connector's SuperSpeed termination transitions from Disabled to Enabled, but there is NOT a USB 2.0 reset

The test may fail if the peripheral device connects over USB 3.0 under these circumstances, because such a device cannot be used to validate this requirement. The device must wait for a USB 2.0 reset before attempting to connect over USB 3.0. For more information about the expected behavior of peripheral devices, see section 10.16.1 of the Universal Serial Bus 3.0 Specification.

Additional Information

Business Justification	The purpose is that the controller manufacturers need to implement UEFI or BIOS handoff per spec. So if system builders configured a system firmware setting to use xhci the controller will follow the configuration appropriately and not switch back to ehci
Enforcement Date	Mar. 01, 2012

System.Fundamentals.TPM.CS

Systems that support Connected Standby must also have a TPM.

Related Requirements	• System.Fundamentals.TPM.CS.ConnectedStandby

System.Fundamentals.TPM.CS.ConnectedStandby

Connected Standby systems must implement TPM v2.0

Target Feature	System.Fundamentals.TPM.CS
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Connected Standby systems must implement a TPM 2.0 solution that meets the requirements called out in **System.Fundamentals.TPM20.TPM20**.

Connected Standby systems must record measurements into PCR [7] as specified in Appendix A of the Microsoft Corporation, "Trusted Execution Environment EFI Protocol, Draft 1.00 dated March 2nd, 2012"[1] document.

[1]This specification must be requested explicitly from Microsoft. To request the current version, please check for its availability on the Microsoft Connect site and if not available, please contact http://go.microsoft.com/fwlink/?LinkId=237130.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.TPM.NonCS

Non CS Devices can have TPM20 or TPM12 but should report themselves as non AOAC devices.

Related Requirements	• System.Fundamentals.TPM.NonCS.NonConnectedStandby

System.Fundamentals.TPM.NonCS.NonConnectedStandby

Requirements for all systems that are not connected standby

Target Feature	System.Fundamentals.TPM.NonCS
Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

All systems that implement TPM1.2 and not connected standby are required to satisfy:

System.Fundamentals.TrustedPlatformModule.TPMRequirements

All systems that implement TPM2.0 and not connected standby are required to satisfy:

System.Fundamentals.TPM20.TPM20

Additional Information

Enforcement Date	Jun. 26, 2013

System.Fundamentals.TPM20

A Trusted Platform Module (TPM) is a microchip designed to provide basic security related functions. Requirements in this area reflect the required TPM version and compatibility with Windows Bitlocker.

Related Requirements	• System.Fundamentals.TPM20.EKCerts • System.Fundamentals.TPM20.TPM20 • System.Fundamentals.TPM20.TPM20Required

System.Fundamentals.TPM20.EKCerts

If the TPM contains a full endorsement KEY certificate, then it must be stored in the TPM NV RAM as described in the TCG PC Client Specific Implementation Specification for Conventional BIOS, section 7.4.5.

Target Feature	System.Fundamentals.TPM20
Applies to	• Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64

Description

If a TPMs contains a full Endorsement Key (EK) certificate stored in the TPM NV RAM it must do so as described in the TCG PC Client Specific Implementation Specification for Conventional BIOS, section 7.4.5: TCG_FULL_CERT. The NV RAM index used for storing the EK certificate must be pre-defined and created with the value of 0x01c00002. If a nonce or template is used to create an EK Certificate which is different from the defaults used by Windows, the same must be specified in the TPM NVRAM under the index of 0x01c00003 for the EK nonce and under the index of 0x01c00004 for the template.

TPM Artifact	NV Index	Required
EK Certificate	0x01c00002	Yes
EK nonce	0x01c00003	Optional, required only if non-default values are used
EK template	0x01c00004	Optional, required only if non-default values are used

The EK certificate must be written to the TPM NVRAM in such a way that the action of Clearing the TPM does not delete the EK certificate.

The certificate must have the EKU specified that indicates that it is indeed an EK Certificate. The OID used for this purpose should be "2.23.133.8.1".

The EK certificate could also be signed using ECDSA. The supported ECC curves for this purpose are NIST 256, 384 and 521. Note that the Endorsement Key is still a RSA 2048 bit key.

The EK certificate must contain an AIA extension that contains the URL for the issuing CA Certificate in the certificate chain. AIA extension (Authority information access locations) must also be present in each non-root cert in the chain with URLs that make the issuing CA certificate (any intermediate CA certs or the root CA cert) – all discoverable and retrievable when starting only with a single EK cert. For more information on AIA extension, please refer to http://technet.microsoft.com/en-us/library/cc753754.aspx.

Note: The EK certificate may be created by the TPM manufacturer or the Platform manufacturer.

Additional Information

Enforcement Date	Jan. 01, 2015

System.Fundamentals.TPM20.TPM20

Requirements for all systems that implement the TPM 2.0 specification

Target Feature	System.Fundamentals.TPM20
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1) • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

For a system that implements TPM 2.0, the platform must comply with the following requirements:

1. The platform shall implement all "Required" features in the table below. "Recommended" entries are for specific configurations.

Integrity Feature	SOC Hardware Functionality	Required	Recommended
Trusted Execution Environment3	***Isolated Storage*** Availability of storage for storing long term secrets. This storage must not be possible to modify by the OS without detection by pre-Operating System components.	X	
	Secure (isolated from runtime OS) storage of: • Values (such as an endorsement primary seed) that survive complete platform power off as well as firmware updates • Values (such as a NV counters) that survive complete platform power off but do not necessarily survive firmware updates (in this case these values shall be	X	

	reset to a random value) • Values (such as the Platform Configuration Registers) that survive platform power-down to the equivalent of ACPI S3 if TPM2_Shutdown(TPM_SU_STATE) is called but may be lost on further power-down.		
Platform Attestation	• Boot measurements recorded in the Platform Configuration Registers for all firmware code loaded after the establishment of the Core Root of Trust for Measurement.	X	
	• Implementation of PCRs 0 through 23 for SHA-1, dedicated to the same boot measurements as TPM 1.2.	X	
	• Support for SHA-1, SHA-256, AES-128, , and RSA-2048 algorithms.	X	
	• Robustness against side channel attacks including Differential Power Analysis (DPA) and Electromagnetic Emanations (EM)		X

2. A platform that does not support a separate, and from the main CPU(s) isolated, cryptographic processing unit must support a Trusted Execution Mode. The Trusted Execution Mode must have a higher privilege level than the Normal Execution Mode, giving it access to data and code not available to the Normal Execution Mode.
3. During the boot sequence, the boot firmware/software shall measure all firmware and all software components it loads after the core root of trust for measurement is established. The measurements shall be logged as well as extended to platform configuration registers in a manner compliant with the following requirements.
4. The measurements must be implemented such that it reliably and verifiably allows a third party to identify all components in the boot process up until the point either the boot finished successfully or when software with a exploited vulnerability was loaded (for example, if the third component loaded includes an exploited vulnerability, then values for the first, second, and third component in the trusted boot log correctly reflect the software that loaded but any values after that may be suspect). To achieve this, the trusted execution environment must provide a mechanism of signing the values of the registers used for Trusted Boot. The interface to the signature ("attestation") mechanism shall comply with the requirements defined in Microsoft Corporation, "Trusted Execution Environment ACPI Profile, 1.0 dated March 2, 2012".
5. The system shall include a trusted execution environment supporting the command set defined in Microsoft Corporation, "TPM v2.0 Command and Signature Profile, 1.0 dated March 2, 2012."
6. The system shall support the interface specified in Microsoft Corporation, "Trusted Execution Environment ACPI Profile, 1.0 dated March 2, 2012".
7. The system shall support the interface and protocol specified in Microsoft Corporation, "Trusted Execution Environment EFI Protocol, 1.0 dated March 2, 21012," with the exception of the PCR[7] measurements specified in Appendix A of that document.
8. The system is required to support measurements into PCR [7] as specified in Appendix A of Microsoft Corporation, "Trusted Execution Environment EFI Protocol,.1.0 dated March 2, 2012" specification. The UEFI firmware update process must also protect against rolling back to insecure firmware versions, or non-production versions that may disable secure boot or include non-production keys. A physically present user may however override the rollback protection manually. In such a scenario (where the rollback protection is overridden), the TPM must be cleared.

   Note: Bitlocker attempts to utilize PCR7 for better user experience and to limit PCR brittleness. If Secure Boot launch of Windows 8 BootMgr requires use of an Allowed DB entry other than the Microsoft-provided EFI_CERT_X509 signature with "CN=Microsoft Windows Production PCA 2011" and "Cert Hash(sha1): 58 0a 6f 4c c4 e4 b6 69 b9 eb dc 1b 2b 3e 08 7b 80 d0 67 8d, then Bitlocker will not be able to utilize PCR7. It is therefore recommended that the only Allowed DB entry for Secure Boot are Microsoft-provided EFI_CERT_X509 signature with "CN=Microsoft

Windows Production PCA 2011" and "Cert Hash(sha1): 58 0a 6f 4c c4 e4 b6 69 b9 eb dc 1b 2b 3e 08 7b 80 d0 67 8d.

9. Platform firmware must ensure invariance of PCRs 0, 2, and 4 across power cycles in the absence of changes to the platform's static core root of trust for measurements (SRTM). Platform firmware must ensure invariance of PCR[7] if implemented as specified in Microsoft Corporation, "Trusted Execution Environment EFI Protocol,.1.0 dated March 2, 2012" across power cycles in the absence of changes to the platform's static core SRTM. Attaching a (non-bootable) USB to the platform or attaching the platform to a docking station shall not cause changes to the SRTM.
10. If the platform does not support clearing the TPM 2.0 endorsement hierarchy (for example, using TPM2_ChangeEPS), an Endorsement Kit (EK) certificate must be provisioned in the TPM NV RAM.
11. Execution of the TPM 2.0 command TPM2_NV_Increment must not require an open object slot.
12. TPM must meet performance requirements as stated below:
    Prior to exit boot services, TPM shall complete extend operations within 20msec. After exit boot services, TPM Extend operations shall complete within 20mS when a TPM is not in a lower power state. If a TPM is in a low power state Extend operations shall complete in 20mS after the TPM has exited its low power state.

The following section is optional when implementing TPM 2.0.

All client SKUs ship TPM 2.0 with SHA2 PCR banks. (Note: It is acceptable to ship TPMs with a single switchable PCR bank that can be utilized for both SHA 1 and SHA 2 measurements.)

All x86/x64 devices equipped with TPM 2.0 can optionally have in the UEFI bios a switch to turn off the TPM device.

Support of ECC algorithms in TPM 2.0 (Note: the TPM library specification requires the support of TPM_ALG_ECDSA and TPM_ALG_ECDH if a TPM implements the TPM_ALG_ECC curve hence are part of requirements) TPM_ECC_NIST_P256 curve should be supported, as specified in Table 8 of TPM library specification Part 2

The following commands for TPM 2.0 to support ECC. These were “recommended” in Windows 8 requirements (details can be found in TCG TPM library specification Part 3)

1. TPM2_ECDH_KeyGen
2. TPM2_ECDH_ZGen
3. TPM2_ECC_Parameters
4. TPM2_Commit

Design Notes

Microsoft uses the term TPM 2.0 for what is also known as TCG TPM.Next.

These requirements may also be met through sealed storage/sealed key blobs that, upon unseal, are held in isolated storage.

The specifications must be requested explicitly from Microsoft. To acquire the current version, first check for its availability on the Microsoft Connect site. If it is not available, contact http://go.microsoft.com/fwlink/?LinkId=237130.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Fundamentals.TrustedPlatformModule

A Trusted Platform Module (TPM) is a microchip designed to provide basic security related functions. Requirements in this area reflect the required TPM version and compatibility with Windows Bitlocker.

Related Requirements	• System.Fundamentals.TrustedPlatformModule.TPMCompliesswithTCGTPMMainSpecification • System.Fundamentals.TrustedPlatformModule.TPMEnablesFullUseThroughSystemFirmware • System.Fundamentals.TrustedPlatformModule.TPMRequirements • System.Fundamentals.TrustedPlatformModule.Windows7SystemsTPM

System.Fundamentals.TrustedPlatformModule.TPMCompliesswithTCGTPMMainSpecification

A system that implements a Trusted Platform Module (TPM) 1.2 must include a TPM that complies with the TCG TPM Main Specification, Version 1.2, Revision 103 (or a later revision), parts 1, 2 and 3.

Target Feature	System.Fundamentals.TrustedPlatformModule
Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64

Description

A system that implements a Trusted Platform Module (TPM) 1.2 must include a TPM that complies with the TCG TPM Main Specification, Version 1.2, Revision 103 (or a later revision), parts 1, 2 and 3. (http://www.trustedcomputinggroup.org/resources/tpm_main_specification)

The TPM must meet the following additional requirements:

- The time required for the TPM to perform all the self-testing performed by the TPM_ContinueSelfTest command must be less than 1 second.
- If the TPM receives a command, after receipt of the TPM_ContinueSelfTest command, but prior to the completion of the TPM_ContinueSelfTest self-test actions, it must not return TPM_NEEDS_SELFTEST.
- The TPM's monotonic counter must be designed to increment at least twice per a platform boot cycle.
- The TPM must implement the TPM_CAP_DA_LOGIC capability for the TPM_GetCapability command.

- By default, the TPM dictionary attack logic must permit at least 9 authorization failures in an a 24 hour time period before entering the first level of defense. Small durations of lockout for less than five seconds are acceptable within a 24 hour period with 9 authorization failures if the TPM leaves the lockout state automatically after five seconds elapses. Alternately, the default system image must contain non-default software anti-hammering settings which correspond to TPM default behavior. (In the Windows 8 OS the settings can be seen by running gpedit.msc then expanding the following in the left tree view: Local Computer Policy\Computer Configuration\Administrative Templates\System\Trusted Platform Module Services. The values to customize are: Standard User Lockout Duration, Standard User Individual Lockout Threshold, and Standard User Total Lockout Threshold.)
- The TPM dictionary attack logic must not permit more than 5000 authorization failures per a year.
- To help platform manufacturers achieve as fast of a boot as possible, the shorter the TPM_ContinueSelfTest execution time, the better.
- It is recommended platform manufacturers provide information about the TPM's dictionary attack logic behavior in customer documentation that includes explicit steps to recover after the TPM enters a locked out state.
- It is recommended the TPM state when shipped is enabled and activated.

Note: Windows uses more TPM functionality than previous releases so Windows Certification Tests for the TPM are more extensive.

Additional Information

Enforcement Date	Mar. 01, 2012

System.Fundamentals.TrustedPlatformModule.TPMEnablesFullUseThroughSystemFirmware

Systems with Trusted Platform Modules enable full use of the TPM including system firmware enhancements

Target Feature	System.Fundamentals.TrustedPlatformModule
Applies to	• Windows 7 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

A system that implements a Trusted Platform Module 1.2 (TPM) must support specific system firmware enhancements. The system firmware code must participate in a measured chain of trust that is established for the pre-operating system boot environment at each power cycle. The system firmware code must support the protected capabilities of the TPM v1.2 and must maintain the SRTM chain of trust.

For a system that implements a Trusted Platform Module (TPM) 1.2, the platform must comply with the following specifications:

- The TCG Platform Reset Attack Mitigation Specification Version 1.00, Revision .92 or later. Revision 1.00 is strong encouraged, including implementation of detecting an orderly OS shutdown(http://www.trustedcomputinggroup.org/resources/pc_client_work_group_platform_reset_attack_mitigation_specification_version_10).
- The TCG Physical Presence Interface Specification Version 1.0 Revision 1.00 or later (http://www.trustedcomputinggroup.org/resources/tcg_physical_presence_interface_specification).
- The TCG PC Client Work Group PC Client Specific TPM Interface Specification (TIS) Version 1.20 Revision 1.00 or later. Implementing version 1.21 or later is strongly encouraged. The TPM must implement the memory mapped space described in the specification. (http://www.trustedcomputinggroup.org/resources/pc_client_work_group_pc_client_specific_tpm_interface_specification_tis)
- If the platform implements UEFI firmware, it must implement
    - The TCG EFI Platform Specification Version 1.20, Revision 1.00 or later, (http://www.trustedcomputinggroup.org/resources/tcg_efi_platform_specification_version_120_revision_10).
    - The TCG EFI Protocol Version 1.20, Revision 1.00 or later, http://www.trustedcomputinggroup.org/resources/tcg_efi_protocol_version_120_revision_10.
- If the platform implements conventional BIOS, it must implement the PC Client Workgroup Specific Implementation Specification for Conventional BIOS, Version 1.20, Revision 1.00 or later. Note: Implementing the errata version 1.21 is strongly recommended. (http://www.trustedcomputinggroup.org/resources/pc_client_work_group_specific_implementation_specification_for_conventional_bios_specification_version_12)
- The TCG ACPI General Specification Version 1.00, Revision 1.00 (http://www.trustedcomputinggroup.org/resources/server_work_group_acpi_general_specification_version_10).
- The Windows Vista BitLocker Client Platform Requirements, dated May 16, 2006, or later, available at http://go.microsoft.com/fwlink/?LinkId=70763. For conventional BIOS systems instead of the PCR measurements listed in the Windows Vista BitLocker Client Platform Requirements, a system may implement the PCR measurements defined in the TCG PC Client Workgroup Specific Implementation Specification for Conventional BIOS, Version 1.21, Revision 1.00.

Design Notes:
The TPM provides a hardware root of trust for platform integrity measurement and reporting. The TPM also provides operating system independent protection of sensitive information and encryption keys.

Additional Information

Exceptions	If a TPM is implemented in the system, then the requirement must be met.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.TrustedPlatformModule.TPMRequirements

System implementing TPM 1.2 must meet requirements

Target Feature	System.Fundamentals.TrustedPlatformModule
Applies to	• Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

For a system that implements a Trusted Platform Module (TPM) 1.2, the platform must comply with the following specifications:

1. The PC Client Work Group Platform Reset Attack Mitigation Specification, Version 1.0 Version 1.00, Revision 1.00 or later, http://www.trustedcomputinggroup.org/resources/pc_client_work_group_platform_reset_attack_mitigation_specification_version_10.
2. The TCG Physical Presence Interface Specification Version 1.2, Revision 1.00, http://www.trustedcomputinggroup.org/resources/tcg_physical_presence_interface_specification.
3. The TCG PC Client Work Group PC Client Specific TPM Interface Specification (TIS) Version 1.21 Revision 1.00 or later (http://www.trustedcomputinggroup.org/resources/pc_client_work_group_pc_client_specific_tpm_interface_specification_tis).
4. If the platform implements UEFI firmware, it must implement
    a. The TCG EFI Platform Specification Version 1.20, Revision 1.0 or later, http://www.trustedcomputinggroup.org/resources/tcg_efi_platform_specification_version_120_revision_10.
    b. The TCG EFI Protocol Version 1.20, Revision 1.00 or later, http://www.trustedcomputinggroup.org/resources/tcg_efi_protocol_version_120_revision_10.
5. If the platform implements conventional BIOS, it must implement the PC Client Workgroup Specific Implementation Specification for Conventional BIOS, Version 1.20, Revision 1.00 or later. Note: The errata version 1.21 is strongly recommended instead. (http://www.trustedcomputinggroup.org/resources/pc_client_work_group_specific_implementation_specification_for_conventional_bios_specification_version_12).
6. The TCG ACPI General Specification Version 1.00, Revision 1.00, http://www.trustedcomputinggroup.org/resources/server_work_group_acpi_general_specification_version_10.
7. Windows Vista BitLocker Client Platform Requirements, dated May 16, 2006, or later (http://go.microsoft.com/fwlink/p/?LinkId=70763). For conventional BIOS systems instead of the PCR measurements listed in the Windows Vista BitLocker Client Platform Requirements, a system may implement the PCR measurements defined in the TCG PC Client Workgroup Specific Implementation Specification for Conventional BIOS, Version 1.21, Revision 1.00.

And the system MUST meet the following additional requirements:

1. Platform firmware must ensure invariance of PCRs 0, 2 and 4 and also PCR 7 if implemented as specified in Appendix A of the Microsoft Corporation, "Trusted Execution Environment EFI Protocol, 1.00 dated March 2nd, 2012" document across power cycles in the absence of changes to the platform's static root of trust for measurements (SRTM). Attaching a (non-bootable) USB to the platform or attaching the platform to a docking station shall not cause changes to the SRTM.
2. If the platform is a server platform, it must reserve PCRs 8 through 15 for OS use. (Note: The same requirement is true for client platforms.)
3. The platform must implement the memory mapped space for the TPM interface (for example, legacy port based I/O is not sufficient).
4. The system firmware must perform Physical Presence Interface operations when the platform is restarted. It is strongly recommended the system firmware performs Physical Presence Interface operations also after shutdown. (Specifically, the Physical Presence Interface Specification Version 1.2, section 2.1.4: Get Platform-Specific Action to Transition to Pre-OS Environment must return a value of 2: Reboot.) (This requirement allows remote administrators to perform Physical Presence Operations without needing to be physically present to turn the platform back on.)
5. The default configuration for the system must have the NoPPIProvision flag specified in the TCG Physical Presence Interface Specification, section 2: Physical Presence Interface set to TRUE.
6. The default system firmware configuration must allow the OS to request Physical Presence operations 6, 7, 10, and 15. Note: The operations are described in Table 2 of the TCG Physical Presence Interface Specification Version 1.2, Revision 1.00.
7. If the system implements the NoPPIClear flag it should do so as specified in the TCG Physical Presence Interface Specification, section 2: Physical Presence Interface. The platform should either provide a system firmware configuration setting to change the flag or implement physical presence operations 17 and 18. Note: The operations and the NoPPIClear flag are described in Table 2 of the TCG Physical Presence Interface Specification Version 1.2, Revision 1.00. (Implementing this flag helps facilitate automated testing of the physical presence interface during Windows certification testing and permits managed environments to completely automate TPM management from the OS without physical presence if an enterprise decides to set the NoPPIClear flag.)
8. The system firmware must implement the _DSM Query method (function index 0) in addition to the Physical Presence Interface methods defined in the TCG Physical Presence Interface Specification, section 2: ACPI Functions. (Please refer to the Advanced Configuration and Power Interface Specification Revision 5.0 for an implementation example of the _DSM Query method.)
9. The system firmware must implement the _DSM Query method (function index 0) in addition to the Memory Clear Interface method defined in the TCG Platform Reset Attack Mitigation Specification, section 6: ACPI _DSM Function. (Please refer to the Advanced Configuration and Power Interface Specification Revision 5.0 for an implementation example of the _DSM Query method.)
10. The system firmware must implement the auto detection of clean OS shutdown and clear the memory overwrite bit as defined in the TCG Platform Reset Attack Mitigation Specification, section 2.3: Auto Detection of Clean Static RTM OS Shutdown. Exception: If the system is able to unconditionally clear memory during boot without increasing boot time, the system may not implement the auto detection (however the pre-boot and ACPI interface implementations are still required).
11. When the system is delivered to an end customer, the TPM permanent flag TPM_PF_NV_LOCKED must be set to TRUE. (This requirement is for systems. For motherboards the flag may be set to FALSE when delivered to the platform manufacturer, however instructions/tools must advise platform manufacturers to set the flag to TRUE before delivery to end customers.)
12. When the system is delivered to an end customer, the TPM permanent flag TPM_PF_NV_ PHYSICALPRESENCELIFETIMELOCK must be set to TRUE. (This requirement is for systems. For motherboards the flag may be set to FALSE when delivered to the platform manufacturer, however instructions/tools must advise platform manufacturers to set the flag to TRUE before delivery to end customers.)
13. The system must contain a full Endorsement Key (EK) certificate stored in the TPM NV RAM as described in the TCG PC Client Specific Implementation Specification for Conventional BIOS, section 7.4.5: TCG_FULL_CERT. The NV RAM index used for storing the EK certificate must be the pre-defined and reserved index TPM_NV_INDEX_EKCert as defined the TPM Main Specification, Part 2, section 19.1.2:

Reserved Index Values. As recommended in the TCG PC Client Specific Implementation Specification for Conventional BIOS, section 4.2.1: TPM Main Specification Reserved Indexes, the D bit attribute must be set for the index. (Note: The certificate may be created by the TPM manufacturer or the platform manufacturer.) Exception: If the system supports generation of a new EK it is not required (but is still strongly recommended) to have an EK certificate.

14. The system firmware must ship with the TPM enumerated by default. (This means the ACPI device object for the TPM must be present by default in the system ACPI tables.)
15. The system firmware must support clearing the TPM from within a setup menu.
16. At least 256 bytes of TPM NVRAM must be reserved (and available) for OS use.
17. The firmware must issue the TPM_ContinueSelfTest during boot such that the self-test completes before the OS loader is launched.
    a. If the TPM device performs the self-test synchronously, the firmware TPM driver should be optimized to issue the command to the device but allow the boot process to proceed without waiting for the return result from the TPM_ContinueSelfTest command. If the firmware TPM driver receives a subsequent command, it should delay the subsequent command until the TPM_ContinueSelfTest command completes instead of aborting the TPM_ContinueSelfTest command.)
    b. A recommendation is to start the self-test before some action which takes at least one second but does not have a dependency on the TPM.
18. The platform may or may not issue the TPM_ContinueSelfTest upon resume from S3 (sleep).
19. The ACPI namespace location for the TPM device object must only depend on the System Bus, ISA or PCI bus drivers provided by Microsoft. The System Bus does not have an ID, but is identified as _SB. The ISA bus device IDs may be PNP0A00 or PCI\CC_0601. The PCI bus device IDs may be PNP0A03 or PCI\CC_0604. In addition, the TPM device object may also depend on these generic bridges, containers or modules: PNP0A05, PNP0A06 and ACPI0004. No other device dependencies are permitted for the ACPI namespace location for the TPM device object.
20. Optional. The platform is recommended to support measurements into PCR [7] as specified in Appendix A of Microsoft Corporation, "Trusted_Execution_Environment_EFI_Protocol, 1.00 dated March 2, 2012" specification, The UEFI firmware update process must also protect against rolling back to insecure firmware versions, or non-production versions that may disable secure boot or include non-production keys. A physically present user may however override the rollback protection manually. In such a scenario (where the rollback protection is overridden), the TPM must be cleared.

Note: Bitlocker utilizes PCR7 for better user experience and limit PCR brittleness. If Secure Boot launch of Windows BootMgr requires use of an Allowed DB entry other than the Microsoft-provided EFI_CERT_X509 signature with "CN=Microsoft Windows Production PCA 2011" and "Cert Hash(sha1): 58 0a 6f 4c c4 e4 b6 69 b9 eb dc 1b 2b 3e 08 7b 80 d0 67 8d, Bitlocker will then not be able to utilize PCR7. It is recommended that the only Allowed DB entry for Secure Boot are Microsoft-provided EFI_CERT_X509 signature with "CN=Microsoft Windows Production PCA 2011" and "Cert Hash(sha1): 58 0a 6f 4c c4 e4 b6 69 b9 eb dc 1b 2b 3e 08 7b 80 d0 67 8d.

"Trusted Execution Environment EFI Protocol, 1.00 dated March 2, 2012" specification must be requested explicitly from Microsoft. To acquire the current version, first check for its availability on the Microsoft Connect site. If it is not available, contact http://go.microsoft.com/fwlink/?LinkId=237130.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Fundamentals.TrustedPlatformModule.Windows7SystemsTPM

Systems with Trusted Platform Modules use TPM family 1.2 Revision 85 or later

Target Feature	System.Fundamentals.TrustedPlatformModule
Applies to	• Windows 7 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

A system that implements a Trusted Platform Module (TPM) must include a TPM that complies with the TPM Main Specification, Version 1.2 Revision 85 (or later), and the TPM Interface Specification, Version 1.2 (or later). In particular, the TPM must implement the memory mapped space required by these specifications. The TPM provides a hardware root of trust for platform integrity measurement and reporting. The TPM also provides operating system independent protection of sensitive information and encryption keys.

Design Notes:
See TCG TPM Specification, Version 1.2, and TCG PC Client TPM Interface Specification, Version 1.2, both available at http://go.microsoft.com/fwlink/?LinkId=58380.

Additional Information

Business Justification	A hardware-based security chip and measurable Static Root of Trust for Measurement (SRTM), provided by the v1.2 TPM along with system firmware with secure features, is a necessary feature that enables Windows boot integrity (an integrity check of core system files) on every boot and restart. In addition, the v1.2 TPM enables stronger, hardware-based protection of sensitive Windows key material. The TPM also enables full volume encryption, which protects the hibernation file, swap files, registry, .INF files, settings, temporary files, and desktop content stored on the fully encrypted volume.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.USBBoot

The feature and requirements are about being able to boot from a USB device.

Related Requirements	• System.Fundamentals.USBBoot.BootFromUSB • System.Fundamentals.USBBoot.SupportSecureStartUpInPreOS

System.Fundamentals.USBBoot.BootFromUSB

System firmware supports booting from all exposed USB 1.x, 2.x, and 3.x ports

Target Feature	System.Fundamentals.USBBoot
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

	• Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

System BIOS or UEFI firmware must by default:

- Support booting through USB 1.x on OHCI controllers
- support booting through USB 2.x on EHCI controllers
- support booting through USB 1.x, 2.x, and 3.x on XHCI controllers.
- Support these on all exposed USB ports up to a hub depth of 3.

The system must also support booting Windows PE images from a USB 2.0 device by using extended INT 13 or UEFI native interface in less than 90 seconds.

Design Notes:
OEMs are encouraged to test the boot functionality by creating a bootable USB flash drive with WinPE. See the OPK for details. Vendors may license WinPE (at no charge). For information, send an e-mail to licwinpe@microsoft.com.

Additional Information

Business Justification	Booting from USB 1.x and 2.x has been a Logo requirement since Windows Vista. Now with USB 3.x entering the market, we need to update this requirement to keep up with modern technologies. Boot from USB is important for BitLocker Recovery as well as booting from WinPE.
Enforcement Date	Jan. 01, 2010

System.Fundamentals.USBBoot.SupportSecureStartUpInPreOS

Systems support secure startup by providing system firmware support for writing to and reading from USB flash devices in the pre-operating system environment

Target Feature	System.Fundamentals.USBBoot
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64 • Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

On all UEFI systems and all systems that implement a TPM, the system must support BitLocker recovery and strong authentication scenarios. This is accomplished by enabling enumeration and full-speed reading/ writing of data (such as key backup and recovery information) from and to a USB mass storage class device, and the UEFI firmware will provide an instance of the block I/O protocol for access to these USB devices.

Design Notes:
See the USB Mass Storage Class Bulk-Only Transport and the USB Mass Storage Class UFI Command specifications, downloadable from http://go.microsoft.com/fwlink/?LinkId=58382.

Additional Information

Business Justification	One of the Secure Startup recovery scenarios requires writing a Recovery Key to a USB flash device and reading it back if the user has to recover the data stored on the encrypted volume on the system hard drive. Also, one of the Secure Startup two-level authentication scenarios requires writing a Startup Key to a USB flash device and reading it back again at system boot time.
Enforcement Date	Mar. 01, 2012

System.Fundamentals.USBDevice

These requirements apply to USB devices that are integrated into a system.

Related Requirements	• System.Fundamentals.USBDevice.SelectiveSuspend

System.Fundamentals.USBDevice.SelectiveSuspend

All internally connected USB devices must support selective suspend by default

Target Feature	System.Fundamentals.USBDevice
Applies to	• Windows 8 Client x86, x64, ARM (Windows RT) • Windows 8.1 Client x86, x64, ARM (Windows RT 8.1)

Description

Selective suspend is an important power saving feature of USB devices. Selective suspension of USB devices is especially useful in portable computers, since it helps conserve battery power.

If a USB device is internally connected, the device driver must enable selective suspend by default. Every USB device driver must place the device into selective suspend within 60 seconds of no user activity. This timeout should be as short as possible while maintaining a good user experience. The selective suspend support can be verified by reviewing the report generated by the powercfg -energy command.

Implementation Notes:
When devices enter selective suspend mode, they are limited to drawing a USB specification defined 2.5mA of current from the USB port. It is important to verify that devices can quickly resume from selective suspend when they are required to be active again.

For example, when selectively suspended, a USB touchpad must detect be able to detect a user's touch and signal resume without requiring the user to press a button. Some devices can lose the ability to detect a wake event when limited to the selective suspend current, 500 microamps per unit load, 2.5mA max. These devices, such as a USB Bluetooth module, must be self-powered, not relying solely on the USB bus for power. By drawing power from another source, the device can detect wake events

For more information about enabling selective suspend for HID devices, please refer to this MSDN article http://msdn.microsoft.com/en-us/library/ff538662(VS.85).aspx

For more information about how to implement selective suspend in a driver, please refer to this white paper: http://www.microsoft.com/whdc/driver/wdf/USB_select-susp.mspx

To specify a port that is internal (not user visible) and can be connected to an integrated device, the ACPI properties _UPC.PortIsConnectable byte must be set to 0xFF and the _PLD.UserVisible bit must be set to 0. More details are available on MSDN.

Additional Information

Business Justification	Many devices, such as fingerprint readers and other kinds of biometric scanners, only require power intermittently. Suspending such devices, when the device is not in use, reduces overall power consumption. More importantly, any device that is not selectively suspended may prevent the USB host controller from disabling its transfer schedule, which resides in system memory. DMA transfers by the host controller to the scheduler can prevent the system's processors from entering deeper sleep states. All connected USB devices must be selective suspended before the host controller can become idle. Thus, it is extremely important that USB devices expected to be connected for prolonged periods of time implement selective suspend correctly, especially embedded devices "inside the plastic."
Enforcement Date	Mar. 01, 2012

System.Fundamentals.WatchDogTimer

A watchdog timer is a device that provides basic watchdog support to a hardware timer exposed by the Microsoft hardware watchdog timer resource table.

Related Requirements	• System.Fundamentals.WatchDogTimer.IfWatchDogTimerImplemented

System.Fundamentals.WatchDogTimer.IfWatchDogTimerImplemented

If a Watch Dog Timer is implemented and exposed through a WDRT (supported for versions prior to Windows 8) or WDAT (required for Windows 8 and later versions), it must meet Windows compatibility and functionality requirements

Target Feature	System.Fundamentals.WatchDogTimer
Applies to	• Windows 7 Client x86, x64 • Windows 8 Client x86, x64

	• Windows 8.1 Client x86, x64 • Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Hardware watchdog timer monitors the OS, and reboots the machine if the OS fails to reset the watchdog The watchdog must meet the requirements and comply with the specification in http://MSDN.microsoft.com/en-us/windows/hardware/gg463320.aspx

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.Base

Basic requirements for server systems

Related Requirements	• System.Server.Base.64Bit • System.Server.Base.BMC • System.Server.Base.BMCDiscovery • System.Server.Base.Compliance • System.Server.Base.DevicePCIExpress • System.Server.Base.ECC • System.Server.Base.Essentials • System.Server.Base.HotPlugECN • System.Server.Base.NoPATA • System.Server.Base.OSInstall • System.Server.Base.PCI23 • System.Server.Base.PCIAER • System.Server.Base.RemoteManagement • System.Server.Base.ResourceRebalance • System.Server.Base.ServerRequiredComponents • System.Server.Base.SystemPCIExpress

System.Server.Base.64Bit

A server system can natively run a 64-bit version of Windows Server

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

A server system must be able to natively support and run a 64-bit Windows Server operating system. Devices in a server system must also have 64-bit drivers available for 64-bit operation.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.Base.BMC

Baseboard management controller solution must meet requirements

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Baseboard management controller (BMC) hardware that uses the Microsoft provided IPMI driver and service provider must comply with the Intelligent Platform Management Interface (IPMI) Specification, Version 1.5 Document (Revision 1.1, February 20, 2002, 6/1/04 MARKUP) or later.

The BMC must be connected to the system through a Keyboard Controller Style (KCS) Interface. The BMC and its KCS interface must be discoverable through ACPI, as prescribed in Appendix C3 of the IPMI V1.5 specification.

Support for interrupt is optional for the BMC. If interrupt is supported, the BMC:

- Must not be shared with other hardware devices.
- Must support the Set Global Flags command.

The BMC driver must support PnP and Power Management according to the minimum device fundamental requirements defined in the Windows Logo Program requirements.

The driver must be compliant to the kernel-mode driver framework (KMDF) component of the Windows Driver Framework (WDF) for the Microsoft Windows family of operating systems. A legacy driver, for backward compatibility reasons, must be Windows Driver Model (WDM) compliant.

The driver must provide a WMI interface, including Timeout Configuration through RequestResponseEx() which is defined in the ipmidrv.mof file of the WMI interface.

The driver must have support for ACPI Control:

- HARDWARE ID - IPI0001
- COMPATIBLE ID - IPI0001 optional
- _SRV - 1.5 or 2.0 IPMI
- _CRS/_PRS - Format of resources: A single 2-byte or two 1-byte each I/O port or memory mapped I/O

The driver must call the Windows ACPI driver to get the above listed ACPI data.

Support for interrupt is optional in the driver, if supported the IPMI driver must:

- Assign only one interrupt
- Not share interrupts
- Handle disabling of non-communication interrupts that the driver does not fully support through the Set Global Flags command.
- Be capable of handling both communication and non-communication interrupts.

Design Notes:
To prevent interrupt storm, the driver enables BMC interrupt when it starts and disables BMC interrupt supports when stops by using the "Set BMC Global Enables" IPMI command. The field needs to set is the bit [0] - Receive Message Queue interrupt. However, this bit is shared for KCS communication interrupt and KCS non-communication, so the driver needs to be able to properly handle both interrupts.
A KCS communication interrupt is defined as an OBF-generated interrupt that occurs during the process of sending a request message to the BMC and receiving the corresponding response. It's also encountered during the course of processing a GET_STATUS/ABORT control code.
A KCS non-communication interrupt is defined as an OBF-generated interrupt that occurs when the BMC is not in the process of transferring message data or getting error status. This will typically be an interrupt that occurs while the interface is in the IDLE_STATE].

Additional Information

Enforcement Date	Jun. 01, 2009

System.Server.Base.BMCDiscovery

Baseboard Management Controller is discoverable and enumerable

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

A system that has a baseboard management controller (BMC) present must expose it for discovery and enumeration by Windows through Plug-and-Play (PnP) methods appropriate for its device interface. If the BMC is connected to the system through a non-PnP legacy link such as the keyboard controller style (KCS) interface, its resources must be exposed through SMBIOS or ACPI for discovery and enumeration by Windows.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.Base.Compliance

Server system includes components and drivers that comply with Windows Hardware Certification Program

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

All buses, devices, and other components in a system must meet their respective Windows Hardware Certification Program requirements and use drivers that are either included with the Windows® operating system installation media or that Microsoft® has digitally signed.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.Base.DevicePCIExpress

Server system includes storage and network solutions that use PCI Express architecture

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

A server system must use PCI Express connectivity for all the storage and network devices installed in the system. The devices may either be adapters installed in PCI Express slots or chip down directly connected to the system board. This requirement does not apply to integrated devices that are part of the Southbridge chipset.

Additional Information

Enforcement Date	Jun. 01, 2008

System.Server.Base.ECC

System memory uses ECC or other technology to prevent single-bit errors from causing system failure

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Server systems must support error correction code, memory mirroring, or another technology that can detect and correct at least a single-bit memory error. The system memory and cache must be protected with ECC) or other memory protection. All ECC or otherwise protected RAM, visible to the operating system must be cacheable. The solution must be able to detect at least a double-bit error in one word and to correct a single-bit error in one word, where "word" indicates the width in bits of the memory subsystem. A detected error that cannot be corrected must result in a system fault.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.Base.Essentials

Windows Storage Server 2008 R2 Essentials system meets requirements.

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Windows Storage Server 2008 R2 Essentials systems must meet the following requirements:

· Flash memory, if present, must meet the following requirements:
· Minimum of 256 MB
· The system must be able to boot from the flash memory
· The flash memory must be updateable
· Up to 2, One Gigabit Ethernet Adapters (10/100/1000baseT PHY/MAC)
· System is required to have two or more USB ports that must at a minimum support the following scenarios:
· Server Recovery and Factory Reset
· Uninterruptable Power Supply (UPS)
· External USB hard drives
NOTE: In order to support the above scenarios, the following features are necessary:
· USB 2.0 functionality must comply with Enhanced Host Controller Interface Specification for Universal Serial Bus 2.0.
· EHCI host controllers must comply with the Enhanced Host Controller Interface Specification.
· USB 2.0 functionality must comply with the power management requirements in USB 2.0 or later.
Recommended: USB 3.0 ports, which if included must have USB 3.0 drivers present in the image and recovery image.

· System must have internal RS-232 port, an internal RS-232 header, or use USB debugging. External RS-232 ports are not allowed.
· System must implement power button as follows:
o When the power button is pressed for less than 4 seconds, the server system must shut down gracefully (in other words, the hardware notifies the software, which initiates a shut down).
o When the power button is pressed for more than 4 seconds, the server system must force a shut down (power off).
· System must have BIOS boot order configured as follows:
o In normal operation, the primary hard drive must be the first boot device.
o In recovery mode, the boot order must be:
· External USB Flash
· USB CD/DVD
· Internal Flash (if implemented)
· Internal DVD/Blu-ray (if HDMI is implemented)
· System must provide Server System Status indicator light
o The Server System Status indicator must use one color, or method, to indicate that the server system is booting and another to indicate that the operating system has booted.
o Each status must be clearly visible

· System can only have an external video connector for HDMI with restricted media playback or for system maintenance that requires video for management. If video out is included in a device that does not include an HDMI interface or is not required for system maintenance, a cap must be provided with the video out disabled.
· When creating the user for enabling HDMI Out User the OEMs should make sure that:
· The HDMI out user password must be randomly generated for each server. Please follow the steps in the HDMI Out document in order to create random password
· It is recommended that the HDMI Out user not be an administrator.
· HDMI output must be configured as follows:
· HDMI Out port must be physically located on the system
· HDMI output can only display a media playback application (once server is booted).
· HDMI output cannot be used to display the server Dashboard or Desktop
· OEM should disable the HDMI Out port until Initial Configuration is complete. This prevents users from seeing the Initial Configuration on HD display
· A remote control must be provided to control the media playback application

· An application must be provided in the server Dashboard that does the following:
· Provide UI for controlling video output resolution, audio, and for controlling video (and music) playback.
· OEMs must configure the HDMI output as the primary display.
· HDMI hardware and software must pass Windows Display Device logo testing.
· DirectX 9 graphics device with WDDM 1.0 or higher driver on the Windows Home Server

· System must meet Windows Logo driver certification requirements for Windows Server 2008 R2.
· System may offer hardware or software based RAID as long as end user management is enabled and system can be recovered in the case of one or more drives failing.

Recommended:
· Systems are allowed to have optical drives as follows:
o DVD and Blu-Ray optical drives allowed for media playback (HDMI) and other applications
o OEMs can choose to ship slot load or tray load optical drives
o Optical drive is required to use SATA
o Optical drive must not have external audio output (front mounted headphone/audio jack)
· Systems may optionally include a wireless network adapter in addition to the required wired NIC. If implemented, the wireless network adapter must meet the following requirements:
o Wireless NIC must be IEEE 802.11 compliant
o Recommended configuration: 802.11n
o No restrictions on antenna design/implementation
o A server Dashboard based wireless configuration application must be provided
o A wired network adapter is required for installation.

Additional Information

Enforcement Date	Jun. 01, 2011

System.Server.Base.HotPlugECN

Server system that supports native Hot Plug functionality meets requirements defined in Hot-Plug ECN No. 31

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

A server system must meet requirements defined in the PCI Hot-Plug ECN No. 31 if it supports hot-plug of PCI Express devices or adapters; for example as an inherent behavior of a dynamically hardware partitionable design, or in the form of either Express Module or a comparable hot-plug PCI Express I/O option design.

Additional Information

Enforcement Date	Aug. 31, 2007

System.Server.Base.NoPATA

Persistent storage devices on servers classified as Hard Disk Drives must not be PATA

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Persistent storage devices classified as Hard Disk Drives, either fixed or removable, must not be controlled by any of the following: Parallel Advanced Technology Attachment (also known as Parallel ATA, PATA, IDE, EIDE, or ATAPI) controllers, to include RAID versions of these devices. PATA controllers of any kind may only be connected to CD, DVD or other storage devices not classified as hard disk drives.

Parallel Advanced Technology Attachment (also known as Parallel ATA, PATA, IDE, EIDE, or ATAPI) controllers, to include RAID versions of these devices, do not support the ability to hot remove a hard disk drive from the system should a hard disk drive fail and need to be replaced. This forces the system to be unavailable for long periods.

Parallel Advanced Technology Attachment (also known as Parallel ATA, PATA, IDE, EIDE, or ATAPI) controllers, to include RAID versions of these devices, do not support the ability to hot remove a hard disk drive from the system should a hard disk drive fail and need to be replaced. This forces the system to be unavailable for long periods.

Additional Information

Enforcement Date	Jun. 01, 2009

System.Server.Base.OSInstall

Server system includes a method for installing the operating system for emergency recovery or repair

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

The server system must provide a method for installing the operating system for emergency repair support. The following are examples of possible solutions:

- PXE support
- Internal or externally attached, bootable, rewriteable DVD.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.Base.PCI23

PCI or PCI-X devices in a server system comply with PCI Local Bus Specification, Revision 2.3 unless otherwise noted

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Server system includes only PCI 2.3 compliant PCI or PCI-X devices unless exemptions are noted by an individual requirement.

Additional Information

Enforcement Date	Jun. 01, 2008

System.Server.Base.PCIAER

Windows Server 8 systems may implement AER (Advanced Error Reporting) as provided by the platform and specified in PCI Express Base Specification version 2.1 and ACPI Specification 3.0b

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

Server System vendors which elect to set the Advanced Error Reporting (AER) bit (0x3) in the PCI _OSC table in the system BIOS must implement the following:

· The _HID value on the root bus of the system must be PNP0A08, so that the operating system can discover the devices are PCI Express and support AER.

· To use PCI AER with Windows the system must report _OSC control in the device and the _SB/PC10 objects

in ACPI. The following bits must be enabled:

· 0x0 - PCI Express Native Hot Plug
· 0x1 - Hot Plug Control
· 0x3 - Advanced Error Reporting (AER)
· 0x4 - PCI Express reliability structure control

· The MCFG ACPI table in PCI Firmware Specification, Revision 3.0b, so that the operating system can access the AER Capability registers in the PCIe extended configuration space.

Design Notes:

This is an If Implemented requirement for Server system vendors. There is no Windows Server 8 requirement to provide AER_OSC to give control to the operating system, and systems may implement a "firmware first" error policy.

Additional Information

Enforcement Date	Jan. 01, 2012

System.Server.Base.RemoteManagement

Server system supports remote, headless, out of band management capabilities

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Server systems must provide the capability of being managed without the operating system being present, or when the operating system is not fully functional.
The system must provide the following remote, headless, out of band management capabilities:

- Power up the server
- Power off the server
- Reset the server
- Provide access to Windows Server Emergency Management Services on the server
- View system Stop errors on the server

The following are not required if the system is a pedestal/standalone system with 8GB or less max RAM that was logo'd for Windows Server 2003 prior to December 31, 2007.

- Change BIOS settings of the server
- Select which operating system to start on the server

The above capabilities can be provided using any combination of the following methods:

- Serial port console redirection
- Service processor
- BIOS redirection
- Baseboard Management Controller
- Other management device

This requirement addresses the minimum capabilities required for headless server support.
Design Notes:

Console redirection can be forced with a setup command-line switch,

[/emsport:{com1|com2|usebiossettings|off}

/emsbaudrate:*baudrate*]

EMS Setup, SPCR and EFI or BIOS redirection settings can be configured per the information at http://msdn2.microsoft.com/en-us/library/ms791506.aspx.

See the Microsoft Headless Server and Emergency Services Design specifications and the IPMI specification at http://go.microsoft.com/fwlink/?linkid=36699.

See service processor console redirection details at http://go.microsoft.com/fwlink/?LinkId=58372.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.Base.ResourceRebalance

Server device drivers must support Resource Rebalance requests

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

A system wide resource rebalance can be executed on Windows Server. One case where this occurs is when a processor is dynamically added to a server. Device drivers must honor the resource rebalance flow and the plug and play requests that are dispatched as part of the flow. Device Drivers must queue all IO requests during the resource rebalance operation.

Additional Information

Enforcement Date	Jun. 01, 2007

System.Server.Base.ServerRequiredComponents

Server system must include necessary devices and functionality

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Server systems must include the following devices or functionality;

Device or Functionality	Requirement	Comments
IO Bus and Devices	System.Server.Base.SystemPCIExpress	PCI Express Root
	System.Fundamentals.SystemPCIController.PCIRequirements	Various PCI Specification reqts
	System.Server.Base.PCI23	Various PCI Specification reqts
	System.Server.Base.HotPlugECN	Hot Plug ECN #31
	System.Server.Base.PCIAER	PCI AER system requirement
	Device.Devfund.Server.PCIAER	PCI AER device requirement
Memory	2008 – 1 TB, 2008 R2- 2TB, 2012 - 4 TB, Server Next - 4TB (subject to revision at RTM)	Maximum supported memory
	512 MB	Minimum supported memory
	System.Server.Base.ECC	ECC
Processor	System.Server.Virtualization.ProcessorVirtualizationAssist	Hyper-V

		support
	1.4 GHz 64-bit	Minimum supported processor speed
	2008 – 64, 2008 R2- 256, 2012 - 640, Server Next - 640 Logical Processors (this is subject to revision at RTM)	Maximum supported processor count
Storage	System.Server.Base.DevicePCIExpress	PCI Express
	System.Server.Base.NoPATA	IDE, EIDE, ATAPI, Parallel ATA not allowed
	System.Fundamentals.Firmware.Boot.SystemWithBootDeviceGreaterThan	No 2.2 TB Boot HD support for BIOS-based systems
Network	Device.Network.LAN.Base	1GigE and minimum off-load requirement
	System.Server.Base.DevicePCIExpress	PCI Express
Install	System.Fundamentals.Firmware.FirmwareSupportsBootingFromDVDDevice	El Torito
Remote Boot	System.Fundamentals.PXE.PXEBoot	PXE or UEFI
Recovery	System.Server.Base.OSInstall	For example: DVD, PXE
Debug	System.Fundamentals.DebugPort.SystemExposesDebugInterface	For example: COM, USB, 1394
Remote & OOB Mngmt	System.Server.Base.RemoteManagement	For example: BMC, Service Processor adapter
High Precision Timer	System.Fundamentals.HAL.HPETRequired	
WHEA	System.Server.WHEA.Core	
SMBIOS	System.Fundamentals.SMBIOS.SMBIOSSpecification	
	System.Server.SMBIOS.SMBIOS	
Video	System.Server.Graphics.WDDM	Use MS-provided driver, or

		provide either Display only or full WDDM driver
	System.Fundamentals.Graphics.MicrosoftBasicDisplayDriver	Video minimums, 1024 x 768 x 32 bits per pixel, VESA timing and compliance
RemoteFX	System.Fundamentals.Firmware.SystemCanBootWithEitherGraphicsAdapter	BIOS support for multiple adapters
	System.Fundamentals.Graphics.MultipleGPUOperatingMode	Functionality of multiple GPUs requirement
Marker file	System.Fundamentals.MarkerFile.SystemIncludesMarkerFile	Marker file for OCA
Single container	System.Client.PCContainer.PCAppearsAsSingleObject	Computer appears as single object in Devices and Printers folder

The following devices or functionality are not required for Server Systems

2. Bus Controllers & Ports
    a. HD Audio
    b. Cardbus & PCMCIA
    c. IEEE 1394
    d. Secure Digital
3. Connectivity
    e. Bluetooth
    f. Cardbus & PCMCIA
    g. ExpressCard
    h. IEEE 1394
    i. Infrared

j. Parallel [sysfund-0221] & Serial

k. Wireless USB

4. Network

l. ISDN

m. TCP Chimney NIC

5. Display

n. Auxiliary Displays

6. Input

o. Smart Card Reader

7. Streaming Media & Broadcast

p. Broadcast Receiver

q. Decoder

r. Encoder

s. Video Capture

8. TPM

t. TPM. If implemented, must meet requirements.

u. USB write for BitLocker Recovery

9. Watchdog Timer (WDT)

10. Baseboard Management Controller (BMC)

11. Enhanced Power Management Additional Qualification

12. Dynamic Partitioning Additional Qualification,

13. Fault Tolerance Additional Qualification

14. High Availability Additional Qualification

15. Power Management concerning S3, S4 and S5 system states support

Additional Information

Enforcement Date	Jun. 26, 2013

System.Server.Base.SystemPCIExpress

Server system supports PCI Express natively

Target Feature	System.Server.Base
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Server systems are required to support a PCI Express root complex as a connection of the I/O system to the CPU and memory must comply with the requirements defined in the PCI Express 1.0a (or 1.1) Base Specification and PCI Local Bus Specification, Revision 2.3. If discrepancies exist, the PCI Express Base Specification takes precedence.

Additional Information

Enforcement Date	Jun. 01, 2008

System.Server.DynamicPartitioning

This feature defines dynamic partitioning requirements of server systems. This feature is not required of all server systems.

Related Requirements	• System.Server.DynamicPartitioning.Application • System.Server.DynamicPartitioning.ApplicationInterface • System.Server.DynamicPartitioning.ConfigurationPersist • System.Server.DynamicPartitioning.Core • System.Server.DynamicPartitioning.ErrorEffect • System.Server.DynamicPartitioning.Firmware • System.Server.DynamicPartitioning.HotAddLocal • System.Server.DynamicPartitioning.HotAddReplace • System.Server.DynamicPartitioning.HotAddVisual • System.Server.DynamicPartitioning.HotReplacePU • System.Server.DynamicPartitioning.PartialHotAdd • System.Server.DynamicPartitioning.SoftwareStatus • System.Server.DynamicPartitioning.Subsystem

System.Server.DynamicPartitioning.Application

Servers that support hardware partitioning must supply partition management software as a Windows application running on a Windows operating system.

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64

	• Windows Server 2012 x64

Description

Servers that support hardware partitioning must provide partition manager software, which provides the user interface administrators will use to configure hardware partitions. This software must be offered as a Windows application running on a Windows operating system.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.ApplicationInterface

Servers that support hardware partitioning must supply partition management software that provides a GUI and a scripting capability for partition management

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Servers that support hardware partitioning must supply partition management software that includes support for a graphical user interface for manual partition management and a scripting capability for remote or automated partition management.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.ConfigurationPersist

Servers that support hardware partitioning must support persistence of hardware partition configuration information across a reboot and power cycle

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

The hardware partition configuration on a server that supports hardware partitioning must persist across a reboot, hibernate, resume, and power cycle of the partition or the server. This requirement assumes that no partition change was initiated while the partition was down.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.Core

Systems that support Dynamic Hardware Partitioning must meet requirements

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Systems must meet the requirements listed below and pass the Dynamic Hardware Partitioning test in the Windows Logo Kit in order to be listed in the Windows Server Catalog as supporting Dynamic Partitioning.

System.Server.DynamicPartitioning.HotAddLocal
System.Server.DynamicPartitioning.ErrorEffect
System.Server.DynamicPartitioning.ConfigurationPersist
System.Server.DynamicPartitioning.Subsystem
System.Server.DynamicPartitioning.PartialHotAdd
System.Server.DynamicPartitioning.HotReplacePU
System.Server.DynamicPartitioning.Application
System.Server.DynamicPartitioning.Firmware
System.Server.DynamicPartitioning.ApplicationInterface
System.Server.DynamicPartitioning.SoftwareStatus
System.Server.DynamicPartitioning.HotAddReplace
System.Server.DynamicPartitioning.HotAddVisual

Additional Information

Enforcement Date	Dec. 01, 2007

System.Server.DynamicPartitioning.ErrorEffect

Errors detected in a hardware partition on servers that support hardware partitioning cause no operating system-detectable effects on other partitions

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Hardware (which includes firmware) or software errors that occur within the boundary of a hardware partition on a server that supports hardware partitioning must not affect the operating system environment within other hardware partitions.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.Firmware

Servers that support hardware partitioning must provide server description and partitioning flows in firmware that comply with the Dynamic Hardware Partitioning Requirements Specification

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

System firmware on a server that supports hardware partitioning provides the ACPI server description, handshaking during partitioning events, and initialization of hardware that is to be added to a partition and must be provided in compliance with the Hot Replace Flow and Requirements and the Hot Add Flow and Requirements specifications.

For access to these specifications, send e-mail to DPFB@Microsoft.com.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.HotAddLocal

Hardware components on a server that supports hardware partitioning that are within a unit that is hot added to a partition cannot be accessible from other hardware partitions

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Processors, memory, and I/O components within any unit that is hot added to an existing hardware partition on a server that supports hardware partitioning must not be directly accessible by software running in any other hardware partition.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.HotAddReplace

Servers that support hardware partitioning must support hot addition of processors, memory, and I/O and hot replace of processor and memory subsystems

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Servers that support hardware partitioning must support hot addition and hot replacement of all operating system-supported component types. Hot-add PU-supported component types are processors, memory, and I/O. Hot replace- supported component types are processors and memory subsystems.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.HotAddVisual

Servers that support hardware partitioning must provide visual user indication of the status of hot-add events if no software-based notification is provided

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Servers that support one or more hot-add component features must provide a visual indication of the status of each hot-add event if no partition management software is provided

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.HotReplacePU

In servers that support dynamic partitioning, hot replacement PUs must have equal and compatible hardware resources to the PU being replaced

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

A processor or memory PU used as a replacement on a server that supports dynamic partitioning must have equal and compatible hardware resources to the PU being replaced; that is, the same processor type and stepping and the same memory configuration.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.PartialHotAdd

Partial success of a hot-add action on a server that supports dynamic partitioning does not affect the stability of the partition or server

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Components associated with a hot-add action on a server that supports dynamic partitioning that fails to start (a parked component) must not have a detrimental effect on other components in the PU, partition, or server.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.SoftwareStatus

Servers that support hardware partitioning must supply partition management software that provides the user with status for each hot-add or hot-replace event

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Servers that support hardware partitioning must supply partition management software. Status of a hot-add or hot-replace event is made available by the Windows operating system in the affected partition. The PM software must provide visual indication of this status to the PM administrator.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.DynamicPartitioning.Subsystem

On servers that support dynamic partitioning, I/O subsystems are provided in a different partition unit to processors and memory subsystems

Target Feature	System.Server.DynamicPartitioning
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

To enable success of the hot replace feature, I/O subsystems must be implemented in a different PU to processors and memory subsystems on servers that support dynamic partitioning.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.FaultTolerant

This feature defines fault tolerant requirements of server systems

Related Requirements	• System.Server.FaultTolerant.Core

System.Server.FaultTolerant.Core

Systems supporting Fault Tolerant operations must meet requirements

Target Feature	System.Server.FaultTolerant
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Systems must meet the requirements listed below and pass the Fault Tolerance test in the Windows Hardware Certification Kit in order to be listed in the Windows Server Catalog as having Fault Tolerance.

A Fault Tolerant set [FT set] of systems is a grouping of systems that provide redundancy for every hardware component in a single system of the FT set and can mask any hardware failure such that network-connected clients are not impacted by the hardware failure, such as by loss of connectivity due to network timeout to the FT set due to the host name, domain name, MAC address or IP address, and the services or applications hosted on the FT set, becoming unavailable to those network connected clients. Additionally, an FT set appears to network-connected clients as one system with a single host name, domain name, MAC address or IP address, and unique instances of services or applications.

An FT set must include system clocks that operate in actual lockstep, i.e., there is only one clock domain for the FT set, or virtual lockstep, i.e., the clocks in the systems that comprise the FT set are synchronized at regular intervals of much less than one second. This allows the FT set to always respond to exactly the same interrupts at exactly

the same time, and thus be executing exactly the same instructions and have exactly the same state at all times, thus providing the required redundancy.

An FT set is able to resynchronize, i.e., make identical, operating system images after a hardware failure in one system of the FT set is corrected, such that network-connected clients are not impacted by the resynchronization, such as by loss of connectivity due to network timeout to the FT set due to the host name, domain name, MAC address or IP address, and the services or applications hosted on the FT set, becoming unavailable to those network connected clients. The correction of the problem may be by replacement or repair of the failed hardware component, or if the hardware failure is transient, may be cleared by a system reset that forces the re-initialization of all the devices in the system that is part of the FT set.

FT systems may disable or not include devices which could cause asynchronous interrupts to occur such that one system in the FT redundant set had to respond to an interrupt to which the other system(s) of the FT set did not experience. Examples of such devices would be monitoring devices [thermal, voltage, etc.], or external devices that would allow a user to inadvertently interrupt or access only one system in an FT set, such as a CD/DVD device, keyboard, mouse, etc.

Additional Information

Enforcement Date	Dec. 01, 2007

System.Server.Firmware.UEFI.GOP

This section describes requirements for systems implementing UEFI firmware.

Related Requirements	• System.Server.Firmware.UEFI.GOP.Display

System.Server.Firmware.UEFI.GOP.Display

System firmware must support GOP and Windows display requirements

Target Feature	System.Server.Firmware.UEFI.GOP
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

If the system firmware supports UEFI, it may chose to additionally support the Graphics Output Protocol (GOP). If the Graphics Output Protocol (GOP) is supported is must be as defined in UEFI 2.3.1.

During this time when the firmware is in control, the following are the requirements:
Topology Selection

- UEFI must reliably detect all the displays that are connected to the POST adapter. The Pre-OS screen can only be displayed on a display connected to the POST adapter.

- In case multiple displays are detected, UEFI must display the Pre-OS screen based on the following logic
    - System with an Integrated display(Laptop, All In One, Tablet): UEFI must display the Pre-OS screen only on the integrated display
    - System **without** an Integrated display (integrated display is shut or desktop system): UEFI must display the Pre-OS screen on one display. UEFI must select the display by prioritizing the displays based on connector type. The prioritization is as follows: DisplayPort, HDMI, DVI, HD15, Component, S-Video. If there are multiple monitors connected using the same connector type, the firmware can select which one to use.

Mode Selection

- Once UEFI has determined which display to enabled to display the Pre-OS screen, it must select the mode to apply based on the following logic
    - System with an Integrated display(Laptop, All In One, Tablet): The display must always be set to its native resolution and native timing
    - System **without** an Integrated display (desktop):
        - UEFI must attempt to set the native resolution and timing of the display by obtaining it from the EDID.
        - If that is not supported, UEFI must select an alternate mode that matches the same aspect ratio as the native resolution of the display.
        - At the minimum, UEFI must set a mode of 1024 x 768
        - If the display device does not provide an EDID, UEDI must set a mode of 1024 x 768
    - The firmware must always use a 32 bit linear frame buffer to display the Pre-OS screen
    - PixelsPerScanLine must be equal to the HorizontalResolution.
    - PixelFormat must be PixelBlueGreenRedReserved8BitPerColor. Note that a physical frame buffer is required; PixelBltOnly is not supported.

Mode Pruning

- UEFI must prune the list of available modes in accordance with the requirements called out in EFI_GRAPHICS_OUTPUT_PROTOCOL.QueryMode() (as specified in the UEFI specification version 2.1)

Providing the EDID

- Once the UEFI has set a mode on the appropriate display (based on Topology Selection), UEFI must obtain the EDID of the display and pass it to Windows when Windows uses the EFI_EDID_DISCOVERED_PROTOCOL (as specified in the UEFI specification version 2.1)to query for the EDID.
- It is possible that some integrated panels might not have an EDID in the display panel itself. In this case, UEFI must manufacture the EDID. The EDID must accurately specify the native timing and the physical dimensions of the integrated panel
- If the display is not integrated and does not have an EDID, then the UEFI does not need to manufacture an EDID

Additional Information

Business Justification	Modern boot experience requires a preboot environment which is both fast and visually appealing. The system UEFI controls the display before Windows takes over the control. This means that the screen controlled by the firmware is the first thing that the user sees. Therefore, it is very important that the user has a great user experience at this stage. Some of the key goals are:Ensure that the screen is visible on exactly one display. Display on a single screen ensures that it is easy for the firmware to set a timing and that the UI is not scaled to fit multiple displays of different sizes and aspect ratios. It is easier for the firmware to display on one display instead of many. The native resolution is important in a number of Windows scenarios:Native resolution provide the sharpest and most clear text. Booting the system in native resolution eliminates the need to change modes during the boot process. The frame buffer can be handed off between bios, boot loader, OS boot, and display driver. The result of this is that the display does not flash during boot and gives a more seamless boot experience. Providing the EDID to Windows is important so that Windows can determine the physical dimensions of the display. Windows will automatically scale its UI to be large on high DPI displays so that the text is large enough for the user to see.
Enforcement Date	Mar. 01, 2012

System.Server.Firmware.VBE

The requirements in this section are enforced on any graphics device with firmware supporting VBE and driver is implementing display portion of the WDDM.

Related Requirements	• System.Server.Firmware.VBE.Display

System.Server.Firmware.VBE.Display

System firmware that supports VBE must comply with the Windows Display requirements

Target Feature	System.Server.Firmware.VBE
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

If a system firmware supports VBE for display control then it must meet the following requirements:
The display is controlled by the video device firmware before the WDDM graphics driver takes over. During this time when the firmware is in control, the following are the requirements:

Topology Selection

- Video device firmware must reliably detect all the displays that are connected to the POST adapter. The Pre-OS screen can only be displayed on a display connected to the POST adapter.
- In case multiple displays are detected, video device firmware must display the Pre-OS screen based on the following logic:

- System with an integrated display(Laptop, All In One, Tablet/Convertible): Video device firmware must display the Pre-OS screen only on the integrated display
- System **without** an integrated display (integrated display is shut or desktop system): Video device firmware must display the Pre-OS screen on one display. The video device firmware must select the display by prioritizing the displays based on connector type. The prioritization is as follows: DisplayPort, HDMI, DVI, HD15, Component, S-Video. If there are multiple monitors connected using the same connector type, the firmware can select which one to use.

Mode Selection

- Once video device firmware has determined which display to enabled to display the Pre-OS screen, it must select the mode to apply based on the following logic
  - System with an Integrated display(Laptop, All In One, Tablet/Convertible): The display must always be set to its native resolution and native timing
  - System **without** an Integrated display (desktop):
    - The video device firmware must attempt to set the native resolution and timing of the display by obtaining it from the EDID
    - If that is not supported, the video device firmware must select an alternate mode that matches the same aspect ratio as the native resolution of the display.
    - At the minimum, the video device firmware must set a mode of 1024 x 768
    - If the display device does not provide an EDID, UEDI must set a mode of 1024 x 768
  - The video device firmware must always use a 32 bit linear frame buffer to display the Pre-OS screen
  - PixelsPerScanLine must be equal to the HorizontalResolution.
  - PixelFormat must be PixelBlueGreenRedReserved8BitPerColor. Note that a physical frame buffer is required; PixelBltOnly is not supported.

Mode Pruning

- The video device firmware must provide a list of modes to Windows when Windows uses the Function 01h (Return VBE Mode Information) as specified in the VESA BIOS Extension Core Functions Standard Version 3.0
- The video device firmware must prune the modes as appropriate. It should only enumerate the modes that are supported in the EDID of the display that is currently active. It is not required to support all the resolutions supported in the EDID
- The video device firmware must support 800 x 600 and 1024 x 768
- All modes must be progress at 60 Hz

Providing the EDID

- Once the video device firmware has set a mode on the appropriate display (based on Topology Selection), video device firmware must obtain the EDID of the display and pass it to Windows when Windows uses command 15h (Display Data Channel) as specified in the VESA BIOS Extension Core Functions Standard Version 3.0
    - It is possible that some integrated panels might not have an EDID in the display panel itself. In this case, video device firmware must manufacture the EDID. The EDID must accurately specify the native timing and the physical dimensions of the integrated panel
    - If the display is not integrated and does not have an EDID, then the video device firmware does not need to manufacture an EDID

Additional Information

Business Justification	The video device firmware controls the display before Windows takes over the control. This means that the screen controlled by the video device firmware is the first thing that the user sees. Therefore, it is very important that the user has a great user experience at this stage. Some of the key goals are: Ensure that the screen is visible on exactly one display. Display on a single screen ensures that it is easy for the firmware to set a timing and that the UI is not scaled to fit multiple displays of different sizes and aspect ratios. It is easier for the firmware to display on one display instead of many. The native resolution is important in a number of Windows scenarios: Native resolution provide the sharpest and most clear text. Booting the system in native resolution eliminates the need to change modes during the boot process. The frame buffer can be handed off between bios, boot loader, OS boot, and display driver. The result of this is that the display does not flash during boot and gives a more seamless boot experience. Providing the EDID to Windows is important so that Windows can determine the physical dimensions of the display. Windows will automatically scale its UI to be large on high DPI displays so that the text is large enough for the user to see.
Enforcement Date	Mar. 01, 2012

System.Server.Graphics

Base for Graphics on Server Systems

Related Requirements	• System.Server.Graphics.WDDM

System.Server.Graphics.WDDM

All Windows graphics drivers must be WDDM

Target Feature	System.Server.Graphics
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

The Windows Display Driver Model (WDDM) was introduced with Windows Vista as a replacement to the Windows XP Display Driver Model (XDDM). The WDDM architecture offers functionality to enable features such as desktop composition, enhanced fault Tolerance, video memory manager, scheduler, cross process sharing of D3D surfaces and so on. WDDM was specifically designed for modern graphics devices that are a minimum of Direct3D 10 Feature Level 9_3 with pixel shader 2.0 or better and have all the necessary hardware features to support the WDDM functionality of memory management, scheduling, and fault tolerance. WDDM for Windows Vista was referred to as "WDDM v1.0". WDDM 1.0 is required for Windows Vista.

Windows 7 made incremental changes to the driver model for supporting Windows 7 features and capabilities and is referred to as "WDDM v1.1" and is a strict superset of WDDM 1.0. WDDM v1.1 introduces support for D3D11, GDI hardware acceleration, Connecting and Configuring Displays, DXVA HD, and other features. WDDM 1.1 is required for Windows 7.

Windows 8 also introduces features and capabilities that require graphics driver changes. These incremental changes range from small changes such as smooth rotation, to large changes such as 3D stereo, and D3D11 video support. The WDDM driver model that provides these Windows 8 features is referred to as "WDDM v1.2" WDDM v1.2 is a superset of WDDM 1.1, and WDDM 1.0.

WDDM v1.2 is required by all systems shipped with Windows 8. WDDM 1.0 and WDDM 1.1 will only be supported with legacy devices on legacy systems. The best experience, and Windows 8 specific features are only enabled by a WDDM 1.2 driver. A WDDM driver that implements some WDDM 1.2 required features, but not all required features will fail to load on Windows 8.

For Windows 8 XDDM is officially retired and XDDM drivers will no longer load on Windows 8 Client or Server.

Windows 8.1 introduces features and capabilities that require graphic driver changes. WDDMv1.3 brings significant improvement in areas related to performance, power and reliability for Windows.

WDDMv1.3 is required by all systems shipped with Windows 8.1.

Below is a summary these WDDM versions:

Operating System	Driver Models Supported	D3D versions supported	Features enabled
Windows Vista	WDDM 1.0 XDDM on Server and limited UMPC	D3D9, D3D10	Scheduling, Memory Management, Fault tolerance, D3D9 & 10
Windows Vista SP1 / Windows 7 client pack	WDDM 1.05 XDDM on Server 2008	D3D9, D3D10, D3D10.1	+ BGRA support in D3D10, D3D 10.1
Windows 7	WDDM 1.1 XDDM on Server 2008 R2	D3D9, D3D10, D3D10.1, D3D11	GDI Hardware acceleration, Connecting and configuring Displays, DXVA HD, D3D11
Windows 8	WDDM 1.2	D3D9, D3D10, D3D10.1, D3D11, D3D11.1	Smooth Rotation, 3D Stereo, D3D11 Video, GPU Preemption, TDR Improvements Diagnostic Improvements, Performance and Memory usage Optimizations, Power Management,

etc.

WDDM v1.2 also introduces new types of graphics drivers, targeting specific scenarios and is described below:

- WDDM Full Graphics Driver: This is the full version of the WDDM graphics driver that supports hardware accelerated 2D & 3D operations. This driver is fully capable of handling all the render, display and video functions. WDDM 1.0 and WDDM 1.1 are full graphics drivers. All Windows 8 client systems must have a full graphics WDDM 1.2 device as the primary boot device.
- WDDM Display Only Driver: This driver is only supported as a WDDM 1.2 driver and enables IHVs to write a WDDM based kernel mode driver that is capable of driving display only devices. The OS handles the 2D or 3D rendering using a software simulated GPU.
- WDDM Render Only Driver: This driver is only supported as a WDDM 1.2 driver and enables IHVs to write a WDDM driver that supports only rendering functionality. Render only devices are not allowed as the primary graphics device on client systems.

Table below explains the scenario usage for the new driver types:

	Client	Server	Client running in a Virtual Environment	Server Virtual
Full Graphics	Required as post device	Optional	Optional	Optional
Display Only	Not allowed	Optional	Optional	Optional
Render Only	Optional as non primary adapter	Optional	Optional	Optional
Headless	Not allowed	Optional	N/A	N/A

Additional Information

Enforcement Date	Jun. 26, 2013

System.Server.Graphics.XDDM

Server Systems with a graphics device implementing a driver based on the XDDM

Related Requirements	• System.Server.Graphics.XDDM.No3DSupport

System.Server.Graphics.XDDM.No3DSupport

Server system graphics solution is based on XDDM unless 3D acceleration is supported

Target Feature	System.Server.Graphics.XDDM
Applies to	• Windows Server 2008 Release 2 x64

Description

A stand alone server system must support the Windows XP display driver model (XDDM) at a minimum. If 3D acceleration is implemented, the driver must be based on the Windows display driver model (WDDM). For more details on how to implement a WDDM based driver, Please refer to the relevant WDK documentation.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Server.PowerManageable

This feature defines power manageable requirements of server systems

Related Requirements	• System.Server.PowerManageable.ACPIPowerInterface • System.Server.PowerManageable.PerformanceStates • System.Server.PowerManageable.RemotePowerControl

System.Server.PowerManageable.ACPIPowerInterface

Power manageable servers support the power metering and budgeting ACPI interface

Target Feature	System.Server.PowerManageable
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Server provides support for reading system level power consumption and reading and writing the system power budget for the server using the 'Power Supply, Metering, and Budgeting Interface' in the ACPI 4.0 specification. The system power budget provides a supported range that the budget can be set to where the minimum budget value is lower than the maximum budget value. The power meter supports a range of averaging intervals such that the minimum averaging interval is one second or lower and the maximum averaging interval is five minutes or higher. To align with the specification, the sampling interval for the power meter must be equal to or less than the minimum averaging interval.

Additional Information

Enforcement Date	Jun. 01, 2009

System.Server.PowerManageable.PerformanceStates

If processor(s) in a server system support performance states, the server provides mechanisms to makes these states available to Windows

Target Feature	System.Server.PowerManageable
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

If the processors on the server support performance states, the server provides firmware mechanisms to pass control of processor performance states to Windows. This mechanism must be enabled by default on the server.

Additional Information

Enforcement Date	Jun. 01, 2006

System.Server.PowerManageable.RemotePowerControl

Power manageable server provides a standards based remote out-of-band interface to query and control the power of the system

Target Feature	System.Server.PowerManageable
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 x86, x64, • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Power manageable server provides an out of band remote management interface from the server's Baseboard Management Controller (BMC) that is complaint with the IPMI, DCMI, or SMASH (via WS-MAN) 'Power State Management Profile' to query the power state, power on or off (soft off) the server remotely.
This is a requirement for the Power Manageable Additional Qualification for Windows Server.

More detail on the SMASH profile can be found on the Distributed Management task Force web site at - http://www.dmtf.org/standards/published_documents/DSP1027.pdf

Additional Information

Enforcement Date	Jun. 26, 2013

System.Server.RemoteFX

This feature defines RemoteFX requirements of server systems

Related Requirements	• System.Server.RemoteFX.RemoteFX

System.Server.RemoteFX.RemoteFX

Server systems supporting RemoteFX must meet requirements

Target Feature	System.Server.RemoteFX
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

Servers must meet the following requirements:

- CPU SLAT- the CPU must support SLAT. This requirement will assist in the performance in the RemoteFX virtualization scenarios.
- GPU requirements - must be WDDM GPUs that support Direct3D11. These requirements apply to only the GPUs intended to support RemoteFX workloads.
- Homogenous GPUs for RemoteFX- workloads - the GPUs that are intended to run RemoteFX workloads must be the same GPU running the same hardware driver.

Additional Information

Enforcement Date	Jun. 26, 2013

System.Server.SMBIOS

This feature defines SMBIOS requirements of server systems

Related Requirements	• System.Server.SMBIOS.SMBIOS

System.Server.SMBIOS.SMBIOS

System firmware must fully and accurately implement SMBIOS structures of type 16 and of type 17

Target Feature	System.Server.SMBIOS

Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

System firmware must fully and accurately implement SMBIOS structures of type 16 (description of physical memory arrays) and of type 17 (description of memory devices), as permitted by the SMBIOS specification. Implementation of other SMBIOS memory description structures - Types 19, 20 and 37 - are optional.
A JEDEC compliant DRAM DIMM supports Serial Presence Detect (SPD). Through this mechanism and defined standards, the module can be identified in terms of its manufacturer, serial number and other useful information. The JEDEC standards require specific data to reside in the lower 128 bytes of an EEPROM located on the memory module. Programming of this EEPROM is normally done by vendors of DRAM DIMMs at their origin of manufacture, and can optionally be redone afterward to meet their OEMs' specifications or retailers' requirements for branding purposes while going through distribution channels.
The system firmware (BIOS or UEFI) probes and extracts this information from the DIMM via its SMBus interface. The system firmware uses this information to configure the memory controller. System firmware that supports SMBIOS V2.4 or later, conveys the above DIMM specific information to the operating systems and running applications via a series of SMBIOS structures ("tables") for memory descriptions. These SMBIOS structures also describe the system memory topology, geometry and characteristics. Those are briefly described here for reference purposes and can be found in the current SMBIOS V2.5 Specification (September 5, 2006):

- Physical Memory Array (Type 16) containing information on Location, Use and Error Correction Types; pages 51-52.
- Memory Array Mapped Address (Type 19) containing address mapping for a Physical Memory Array (one structure is present for each contiguous address range described); page 56.
- Memory Device (Type 17) containing information on Form Factor, Type and Type Detail; pages 52-54
- Memory Device Mapped Address (Type 20) containing address mapping to a device-level granularity (one structure is present for each contiguous address range described); page 56.
- Memory Channel (Type 37) containing correlation between a Memory Channel and its associated Memory Devices (each device presents one or more loads to the channel); page 68. This support in the system firmware will:
    - Allow the customers to manage their server memory components as deployed IT assets, and to maintain a comprehensive understanding of their investment of these assets in terms of RAS abilities and cost of ownership.
    - Allow server and data center management solutions to exploit this information in their diagnostic tools and methods for better RAS abilities.
    - Enable certain classes of ISV products (RAM disk, etc.) to exploit this information for better performance and functionalities on Windows platforms.

Additional Information

Enforcement Date	Jun. 01, 2009

System.Server.SVVP

This feature defines requirements for the SVVP program

Related Requirements	• System.Server.SVVP.SVVP

System.Server.SVVP.SVVP

Products participating in the Server Virtualization Validation Program must meet requirements

Target Feature	System.Server.SVVP
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2012 x64

Description

Server platforms participating in the Server Virtualization Validation Program must meet the requirements called out here: http://www.windowsservercatalog.com/svvp.aspx.

Additional Information

Enforcement Date	Jan. 01, 2011

System.Server.SystemStress

This feature defines system stress requirements of server systems

Related Requirements	• System.Server.SystemStress.ServerStress

System.Server.SystemStress.ServerStress

Server system must function correctly under stress

Target Feature	System.Server.SystemStress
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Server system must operate correctly under long-haul, non-deterministic, high stress conditions. The hardware and software components of the Server system must not cause data corruption, hangs, leaks, memory resource fragmentation, crashes, or have impacts on other components of the system. Server systems must be able to

reliably shutdown and restart while under stress to prevent unnecessary and unplanned downtime.
This will be tested using stress tools that emulate loads which may be placed upon a Windows Server system

Additional Information

Enforcement Date	Jan. 01, 2011

System.Server.Virtualization

This feature defines virtualization requirements of server systems

Related Requirements	• System.Server.Virtualization.ProcessorVirtualizationAssist

System.Server.Virtualization.ProcessorVirtualizationAssist

Processors in the server support virtualization hardware assists

Target Feature	System.Server.Virtualization
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

All processors in the server must support one of the following processor virtualization hardware assist technologies:

- Intel VT technology
- AMD-V technology

Details on specific requirements for each of these technologies are available in the Windows Server 2008 Virtualization Requirements document.

For access to the Windows Server 2008 Virtualization Requirements document, send e-mail to lhvrtreq@microsoft.com.

Additional Information

Enforcement Date	Jun. 01, 2007

System.Server.WHEA

This feature defines WHEA requirements of server systems

Related Requirements	• System.Server.WHEA.Core

System.Server.WHEA.Core

Server enables reporting of system hardware errors to the operating system

Target Feature	System.Server.WHEA
Applies to	• Windows Server 2012 R2 x64 • Windows Server 2008 Release 2 x64 • Windows Server 2012 x64

Description

Servers are required to provide mechanisms to enable reporting or communication of corrected and uncorrected system hardware errors that are available on the server to the operating system. The platform may perform thresholding of corrected errors.

The minimal set of error sources are:
* IA64 - Machine Check Exception, Corrected Machine Check, Corrected Platform Error, INIT, PCI Express AER
* X86-64 - Machine Check Exception, Corrected Machine Check, Non Maskable Interrupt, PCI Express AER, BOOT errors.

An interface must be provided on the server to facilitate persistence of error records. The interface must preserve the error records across a server reboot and power cycle. At a minimum, the platform must provide enough storage space for one uncorrectable error record. Options to implement this interface are described in the WHEA Platform Design Guide.

Windows Server provides an OSC mechanism to indicate the presence of WHEA in the OS. The server must honor these mechanisms and enable WHEA flows when the OSC is detected. Optional mechanisms are provided to enable the firmware to process error events from specified error sources before handing control off to WHEA and to communicate this behavior to the OS. This helps avoid conflicts on software handling of hardware error events. These mechanisms are described in the WHEA Platform Design Guide.

Servers must support WHEA-defined interfaces for software insertion of a subset of hardware error conditions into the platform to enable WHEA validation. The injection mechanism must support the injection of one fatal uncorrected and one corrected error; each injectable error is injected using one of the error sources on the platform, and using the signaling mechanism specified for that error source. Options to provide this interface are described in the WHEA Platform Design Guide.

For access to the WHEA Platform Design Guide, send e-mailto WHEAFB@Microsoft.com.

Additional Information

Enforcement Date	Jun. 01, 2009