Windows Hardware Certification Requirements

Client and Server Systems

December 2011

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.

Introduction

This release to web (RTW) document contains the Windows Hardware Certification requirements for Windows 8 Certified Systems. These requirements are Microsoft's guidelines for designing systems which successfully meet Windows performance, quality, and feature criteria, to assure the optimum Windows 8 computing experience. Successfully following this guidance will allow a partner to receive certification for their system.

The requirements are organized by feature using a Camel Case naming convention, which facilitates grouping related requirements and communicating their relationship to the Windows feature they are intended to support. Tests assessing compliance with the features are exposed during testing with the Hardware Certification Kit and can be related directly back to these requirements.

Some requirements have passed forward from Logo requirements for earlier Windows versions which used a category based structure. We have included the older LogoPoint ID in the comments section for your convenience.

Certified Devices in Systems

A Windows Certified System is composed of Windows certified components. These internal devices are certified before system integration, leaving the focus of system testing on those features which require the coordination of multiple components or configuration settings unique to the system under test. At the system certification level, if the system includes a device or component, then all the requirements associated with that device as exposed to Windows must be met. We encourage a review of the requirements for each system components incorporated into the design of a system.

If Implemented Requirements

The Windows Hardware Certification program declares requirements which must be met by any system under the feature area of system fundamentals. Additional functionality is built into Windows which is optional, and can offer a competitive edge for manufacturers should they chose to implement these features. In these cases, there are requirements which must be met only if the additional functionality is implemented. Because these additional requirements apply only if the relevant functionality is implemented, they are referred to as "if-implemented" requirements in this document.

The Hardware Certification Kit detects features exposed by a product automatically. The detected features the will be tested for compliance whether they were mandatory to successfully be certified as a defined product type or if the feature is optional. The title of the requirement or the exception field will indicate when a requirement applies.

Features

System.Client.Aero

Description:

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

Target Feature: System.Client.Aero

Title: Systems are capable of starting the Aero theme

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client 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.

Exceptions:

None

Business Justification:

This is required for supporting the Desktop Windows Manager and enabling Aero.

Scenarios:

Enables Aero and Desktop Windows Manager

Success Metric: Pass/Fail

Enforcement Date: June 1, 2007

Comments:

SYSFUND-0046

System.Client.BluetoothController.Base

Description:

These requirements apply to systems that have generic Bluetooth controllers

Related Requirements:

- System.Client.BluetoothController.Base.4LeSpecification
- System.Client.BluetoothController.Base.Soc
- System.Client.BluetoothController.Base.SystemsWithBluetoothImplementDeviceID

System.Client.BluetoothController.Base.4LeSpecification

Target Feature: System.Client.BluetoothController.Base

Title: If a system includes a Bluetooth controller it must support the Bluetooth 4.0+LE specification requirements

Applicable OS Versions:

- Windows 8 Client x64
- Windows 8 Client x86
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification: Not Specified

Scenarios: Not Specified

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

BUSPORT-8002

System.Client.BluetoothController.Base.CS

Target Feature: System.Client.BluetoothController.Base

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

This ensures good Windows user experience for connected standby platforms.

Scenarios:

Systems that support connected standby.

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

New

System.Client.BluetoothController.Base.SystemsWithBluetoothImplementDeviceID

Target Feature: System.Client.BluetoothController.Base

Title: Systems which support Bluetooth must implement the DeviceID profile, version 1.4

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Systems which support Bluetooth must include the Device ID record as specified in the DeviceID profile, version 1.4. This record shall contain the devices VID/PID and ContainerID.

Exceptions: Not Specified

Business Justification:

The expectation is that when connecting a PC to another PC over Bluetooth, the system would appear as a single device in the Devices and printers folder.

Scenarios:

Great with devices, the user can easily see, configure and use all devices from one place.

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8011

System.Client.BluetoothController.NonUSB

Description:

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

Target Feature: System.Client.BluetoothController.NonUSB

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Any platform using a non-USB connected BT controller must ship with MSFTs inbox BT stack

Exceptions: Not Specified

Business Justification:

This ensures a good Windows user experience.

Scenarios: Not Specified

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

New

System.Client.BluetoothController.NonUSB.ScoSupport

Target Feature: System.Client.BluetoothController.NonUSB

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

This ensures a good Windows audio user experience.

Scenarios: Not Specified

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

New

System.Client.BrightnessControls

Description:

This section describes requirements systems with brightness controls.

Related Requirements:

- System.Client.BrightnessControls.BrightnessControlButtons

System.Client.BrightnessControls.BrightnessControlButtons

Target Feature: System.Client.BrightnessControls

Title: 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

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

The brightness control slider will update when the end user changes the screen brightness with hardware key buttons.

Scenarios:

Change brightness of the monitor

Success Metric: Pass/Fail

Enforcement Date: Windows 7 RC

Comments:

SYSFUND-0183

System.Client.Digitizer.Base

Description:

Base feature for digitizer

Related Requirements:

- System.Client.Digitizer.Base.DigitizersAppearAsHID
- System.Client.Digitizer.Base.HighQualityDigitizerInput
- System.Client.Digitizer.Base.HighQualityTouchDigitizerInput

System.Client.Digitizer.Base.DigitizersAppearAsHID

Target Feature: System.Client.Digitizer.Base

Title: Please refer to Device.Digitizer.DigitizersAppearAsHID

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Base.DigitizersAppearAsHID

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

System.Client.Digitizer.Base.HighQualityDigitizerInput

Target Feature: System.Client.Digitizer.Base

Title: Please refer to Device.Digitizer.Base.HighQualityDigitizerInput

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Base.HighQualityDigitizerInput

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

INPUT-0001v5

System.Client.Digitizer.Base.HighQualityTouchDigitizerInput

Target Feature: System.Client.Digitizer.Base

Title: Please refer to Device.Digitizer.Base.HighQualityTouchDigitizerInput

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Base.HighQualityTouchDigitizerInput

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

INPUT-0046v8

System.Client.Digitizer.Pen

Description:

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

Target Feature: System.Client.Digitizer.Pen

Title: Please refer to Device.Digitizer.Pen.100HzSampleRate

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Pen.100HzSampleRate

Scenarios:

Pen

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Port of Win7 pen requirement

System.Client.Digitizer.Pen.ContactAccuracy

Target Feature: System.Client.Digitizer.Pen

Title: Please refer to Device.Digitizer.Pen.ContactAccuracy

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Pen.ContactAccuracy

Scenarios:

Pen

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Port of Win7 pen requirement

System.Client.Digitizer.Pen.HoverAccuracy

Target Feature: System.Client.Digitizer.Pen

Title: Please refer to Device.Digitizer.Pen.HoverAccuracy

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Pen.HoverAccuracy

Scenarios:

Pen

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Port of Win7 pen requirement

System.Client.Digitizer.Pen.PenRange

Target Feature: System.Client.Digitizer.Pen

Title: Please refer to Device.Digitizer.Pen.PenRange

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Pen.PenRange

Scenarios:

Pen

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 15

System.Client.Digitizer.Pen.PenResolution

Target Feature: System.Client.Digitizer.Pen

Title: Please refer to Device.Digitizer.Pen.PenResolution

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Pen.PenResolution

Scenarios:

Pen

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Port of Win7 pen requirement

System.Client.Digitizer.Touch

Description:

Windows Touch interface for digitizer devices.

Related Requirements:

- System.Client.Digitizer.Touch.5TouchPointMinimum
- System.Client.Digitizer.Touch.Bezel
- 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.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

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.5TouchPointMinimum

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.5TouchPointMinimum

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 1

System.Client.Digitizer.Touch.Bezel

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.Bezel

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.Bezel

Scenarios:

Please refer to Device.Digitizer.Touch.Bezel

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.Digitizer.Touch.DigitizerConnectsOverUSBOrI2C

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.DigitizerConnectsOverUSBOrI2C

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.DigitizerConnectsOverUSBOrI2C

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 3; Update

System.Client.Digitizer.Touch.DigitizerJitter

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.DigitizerJitter

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.DigitizerJitter

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 7

System.Client.Digitizer.Touch.ExtraInputBehavior

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.ExtraInputBehavior

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.ExtraInputBehavior

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 4; Updated

System.Client.Digitizer.Touch.FieldFirmwareUpdatable

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.FieldFirmwareUpdatable

Applicable OS Versions:

- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Client x86

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.FieldFirmwareUpdatable

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

New

System.Client.Digitizer.Touch.HIDCompliantFirmware

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.HIDCompliantFirmware

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.HIDCompliantFirmware

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 13; Update

System.Client.Digitizer.Touch.HighResolutionTimeStamp

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.HighResolutionTimeStamp

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.HighResolutionTimeStamp

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

New

System.Client.Digitizer.Touch.InputSeparation

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.InputSeparation

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.InputSeparation

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 10; Updated

System.Client.Digitizer.Touch.NoiseSuppression

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.NoiseSuppression

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.NoiseSuppression

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 8

System.Client.Digitizer.Touch.PhysicalDimension

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.PhysicalDimension

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.PhysicalDimension

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 16; Updated

System.Client.Digitizer.Touch.PhysicalInputPosition

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.PhysicalInputPosition

Applicable OS Versions:

- Windows 8 Client x86

- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.PhysicalInputPosition

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 14; Update

System.Client.Digitizer.Touch.PowerStates

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.PowerStates

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.PowerStates

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 12; Update

System.Client.Digitizer.Touch.ReportingRate

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.ReportingRate

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.ReportingRate

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 9

System.Client.Digitizer.Touch.ResponseLatency

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.ResponseLatency

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.ResponseLatency

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 11; Updated

System.Client.Digitizer.Touch.TouchResolution

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.TouchResolution

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.TouchResolution

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

Touch 6

System.Client.Digitizer.Touch.ZAxisAllowance

Target Feature: System.Client.Digitizer.Touch

Title: Please refer to Device.Digitizer.Touch.ZAxisAllowance

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

Please refer to Device.Digitizer.Touch.ZAxisAllowance

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

New

System.Client.Firewall

Description:

The requirements in this section describe what is required.

Related Requirements:

- System.Client.Firewall.FirewallEnabled

System.Client.Firewall.FirewallEnabled

Target Feature: System.Client.Firewall

Title: Systems enable a firewall solution by default

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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

Exceptions:

None

Business Justification:

End users have an expectation their systems are secure out of the box. The inbox Windows Firewall must be enabled if there is no other firewall applications installed on the system.

Scenarios:

Having a firewall enable provides a layer of protection for the end user.

Success Metric: Pass/Fail

Enforcement Date: Windows 7 RC

Comments:

SYSFUND-0204

System.Client.Firmware.UEFI

Description:

This section describes requirements for systems implementing UEFI firmware.

Related Requirements:

- System.Client.Firmware.UEFI.GOP.Display

System.Client.Firmware.UEFI.GOP.Display

Target Feature: System.Client.Firmware.UEFI

Title: System firmware must support GOP and Windows display requirements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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

1. 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.
2. In case multiple displays are detected, UEFI must display the Pre-OS screen based on the following logic
    a. System with an Integrated display(Laptop, All In One, Tablet): UEFI must display the Pre-OS screen only on the integrated display
    b. 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

1. 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
    a. System with an Integrated display(Laptop, All In One, Tablet): The display must always be set to its native resolution and native timing
    b. System **without** an Integrated display (desktop):
        i. UEFI must attempt to set the native resolution and timing of the display by obtaining it from the EDID.
        ii. If that is not supported, UEFI must select an alternate mode that matches the same aspect ratio as the native resolution of the display.
        iii. At the minimum, UEFI must set a mode of 1024 x 768
        iv. If the display device does not provide an EDID, UEFI must set a mode of 1024 x 768
    c. The firmware must always use a 32 bit linear frame buffer to display the Pre-OS screen
    d. PixelsPerScanLine must be equal to the HorizontalResolution.

e. PixelFormat must be PixelBlueGreenRedReserved8BitPerColor. Note that a physical frame buffer is required; PixelBltOnly is not supported.

Mode Pruning

1. 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

1. 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.

    a. 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

    b. If the display is not integrated and does not have an EDID, then the UEFI does not need to manufacture an EDID

Exceptions: Not Specified

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.

Scenarios:

1. Power up a laptop/portable device
    a. In this case, the firmware will determine the native resolution of the integrated display and display the pre-OS screen at the native resolution
2. Power up a desktop with multiple monitors connected
    a. In this case, the firmware will determine the best display to enable and then use the native resolution of the display
3. In case the 3rd party WDDM driver is disabled, the, Microsoft Basic Display Driver will be running. This driver will only allow the user to use the resolution that the system was in before the Microsoft Basic Display driver was run.

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8108 For client systems BIOS mode is only allowed for systems that ship with 32 bit Windows. This allows OEMs to maintain one firmware image for both 64 bit and 32 bit deployments. All systems must be UEFI mode capable and also pass 64 bit UEFI logo requirements.

System.Client.Graphics

Description:

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

Related Requirements:

- System.Client.Graphics.FullGPU
- System.Client.Graphics.NoMoreThanOneInternalMonitor
- System.Client.Graphics.WDDM
- System.Client.Graphics.WDDMSupportRotatedModes
- System.Client.Graphics.Windows7.MinimumDirectXLevel

System.Client.Graphics.FullGPU

Target Feature: System.Client.Graphics

Title: A Windows client system must have a “Full” graphics device and that device must be the post device.

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification: Not Specified

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

New; Sysfund-8501

System.Client.Graphics.NoMoreThanOneInternalMonitor

Target Feature: System.Client.Graphics

Title: Graphics driver must not enumerate more than one monitor as internal

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 7 Client x64
- Windows 7 Client x86
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

Having two displays marked as internal is not supported.

Scenarios:

Being able to project the screen to an externally connected monitor.

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8013

System.Client.Graphics.WDDM

Target Feature: System.Client.Graphics

Title: All Windows graphics drivers must be WDDM

Applicable OS Versions:

- Windows 8 Server x64
- Windows 8 Client ARM
- Windows 8 Client x64
- Windows 8 Client x86

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.

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:

a. 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.

b. 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.

c. 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

Exceptions: Not Specified

Business Justification:

See description

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

New; Sysfund-8096

System.Client.Graphics.WDDMSupportRotatedModes

Target Feature: System.Client.Graphics

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

Applicable OS Versions:

- Windows 8 Client ARM
- Windows 8 Client x64
- Windows 8 Client x86

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.

Exceptions: Not Specified

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.

Scenarios:

User takes a new Windows 8 device and rotates the device from landscape to portrait mode

Due to the presence of the accelerometer Windows is able to automatically adjust the desktop screen to match the new orientation

There should be no cases where the driver does not support this rotation

Alternately, the user can also go to the Display Control Panel and rotate the desktop

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

New;

System.Client.Graphics.Windows7.MinimumDirectXLevel

Target Feature: System.Client.Graphics

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

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client 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

Exceptions: Not Specified

Business Justification:

The goal of this requirement is create a common developer platform for Windows 7 systems.

Scenarios:

Developer Experience.

Success Metric: Pass/fail

Enforcement Date: December 1, 2009

Comments:

Graphics-0020 and SYSFUND-0192

System.Client.MediaTranscode

Description:

Requirements describe the capabilities that are required for media transcode.

Related Requirements:

- System.Client.MediaTranscode.SystemTranscodeFasterThanRealTime

System.Client.MediaTranscode.SystemTranscodeFasterThanRealTime

Target Feature: System.Client.MediaTranscode

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

Consumers increasingly expect their PCs to provide the quality & reliability typically provided by dedicated consumer electronics devices. These requirements help ensure that Windows delivers the high quality transcode experiences that consumers have come to expect with scenarios such as:

- Play To Home Media Sharing
- Sync to Portable/Mobile Devices
- Webcam Capture
- Movie Making/Editing Apps

Scenarios:

Transcode high definition video.

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8014

System.Client.MobileBroadBand

Description:

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

Related Requirements:

- System.Client.MobileBroadBand.ClassDriver

System.Client.MobileBroadBand.ClassDriver

Target Feature: System.Client.MobileBroadBand

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

USB based GSM and CDMA class of Mobile Broadband device firmware implementation must comply with the Microsoft's Mobile Broadband Interface Model Specification. No additional IHV drivers are needed for the functionality of the device and should work with Microsoft's generic class driver implementation. Note that Microsoft's class driver does not support WiMax class of devices and a separate IHV driver is needed for WiMax devices.

Exceptions: Not Specified

Business Justification:

Better user experience and increased quality of the Mobile Broadband devices.

Scenarios: Not Specified

Success Metric: Passing Mobile Broadband logo tests in WHCK. Mobile Broadband USB Devices for GSM, CDMA and WiMAX technologies covering embedded and dongle form-factor devices. Embedded modules using PCI Express Mini Card Form Factors (both half size as well as full-size) and providing USB interfaces also qualify as USB devices category.

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.NearFieldProximity

Description:

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.NFCPlacement
- System.Client.NearFieldProximity.RangeOfActuation
- System.Client.NearFieldProximity.TouchMark

System.Client.NearFieldProximity.ImplementingProximity

Target Feature: System.Client.NearFieldProximity

Title: How/when to implement NFP technology

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

Ensures a consistent and confident user experience across proximity enabled systems. _

Scenarios:

Tap against an NFC only system and a proximity experiences should work

Success Metric: Pass/Fail

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.NearFieldProximity.NFCPlacement

Target Feature: System.Client.NearFieldProximity

Title: Placement of NFC on all form factors

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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). Refer to the most current version of the 'Windows 8 Near Field Proximity Implementation Specification' document for placement guidance and mark usage requirements. The spec can be found at:

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

Exceptions: Not Specified

Business Justification:

Ensures a successful user experience and consistent tap interaction between windows devices. _

Scenarios:

When tapping two devices together with two tablets, both tablets should line up the radios with each other (meaning the upper right quadrants are lined up) and then a successful tap will occur.

Success Metric: Pass/fail

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.NearFieldProximity.RangeOfActuation

Target Feature: System.Client.NearFieldProximity

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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

Exceptions:

None

Business Justification:

Ensures connections will only be established in a small range rather than larger ones (such as Bluetooth or wireless). Confines the range so that the appropriate technology is tested._

Scenarios:

When tapping two devices together, a connection cannot be established if the touch point of the host device is greater than 10 cm away from the other touch point.

Success Metric: Pass/Fail

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.NearFieldProximity.TouchMark

Target Feature: System.Client.NearFieldProximity

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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). Refer to the most current version of the 'Windows 8 Near Field Proximity Implementation Specification' document for placement guidance and mark usage requirements. The spec can be found at:

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

Exceptions: Not Specified

Business Justification:

Users will know where to tap devices together

Scenarios:

Users want to have a proximity experience and need to know where to tap devices together.

Success Metric: Pass/Fail

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.PCContainer

Description:

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

Target Feature: System.Client.PCContainer

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

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows Server 2008 Release 2 x64
- Windows 8 Server x64

Description:

Computers must appear as a single object in the Windows user interface (UI), such as 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 UI. 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 and other UI elements.

The computer must appear as a single device container in all Windows UI elements for the following reason:

- The Windows UI 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. Users will interact with the computer and connected devices using this device centric UI. For example, 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, system firmware, 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 Hardware Certification 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 WHCK. 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

Exceptions: Not Specified

Business Justification:

The Windows user interface 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.

Scenarios:

The Windows user interface provides and a logical and understandable representation of a PC.

Success Metric: Not Specified

Enforcement Date: Windows 7 RC

Comments:

SYSFUND-0200

System.Client.RadioManagement

Description:

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

Target Feature: System.Client.RadioManagement

Title: 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

Applicable OS Versions:

- Windows 8 Client ARM
- Windows 8 Client x64
- Windows 8 Client x86

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:

1. Toggle Button (Laptops and Tablets)
2. Toggle Button with LED (Non-Connected standby supported laptops and tablets)
3. A-B slider switch (Laptops and Tablets)
4. 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.

Exceptions: Not Specified

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.

Scenarios:

User has a consistent predictable way to turn off all wireless capabilities on the PC

Success Metric: The certification tests will check that the appropriate HID message is sent and received, the radio management UI is updated to reflect the correct new state of the hardware button and when the user changes the state in the UI, a HID message must be sent to the LED and its state must be updated.

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.RadioManagement.RadioMaintainsState

Target Feature: System.Client.RadioManagement

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

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.

Scenarios:

For airplane mode the user has a consistent predictable way to turn off all wireless capabilities on the PC. User has granular radio management. They can turn off/on individual radios to manage their power, privacy settings or to manage bandwidth usage. Also, on airplanes that support Wi-Fi, users can turn on Wi-Fi while still blocking other radios from broadcasting, and conserve battery power.

Success Metric: Pass/Fail

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.RadioManagement.RadioManagementAPIHID

Target Feature: System.Client.RadioManagement

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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 changes the state in all radios present in the PC and it communicates the change of state to the Radio Management API using the HID driver.

The HID usage IDs are:

Usage ID	Usage Name	Usage Type
0A	Wireless Radio Controls	CA
C0	Wireless Radio Button	OOC

C1	Wireless Radio LED	OOC
C2	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 0A

COLLECTION (Application) A1 01

PHYSICAL_MINIMUM (0) 35 00

PHYSICAL_MAXIMUM (1) 45 01

LOGICAL_MINIMUM (0) 15 00

LOGICAL_MAXIMUM (1) 25 01

USAGE (Wireless Radio Button) 09 C0

REPORT_COUNT (1) 95 01

REPORT_SIZE (1) 75 01

INPUT (Data,Var,Rel) 81 06

REPORT_COUNT (1) 95 01

REPORT_SIZE (7) 75 07

INPUT (Cnst,Var,Rel) 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 0A

COLLECTION (Application) A1 01

PHYSICAL_MINIMUM (0) 35 00

PHYSICAL_MAXIMUM (1) 45 01

LOGICAL_MINIMUM (0) 15 00

LOGICAL_MAXIMUM (1) 25 01

USAGE (Wireless Radio Button) 09 C0

REPORT_COUNT (1) 95 01

REPORT_SIZE (1) 75 01

INPUT (Data,Var,Rel) 81 02

REPORT_SIZE (7) 75 07

INPUT (Cnst,Var,Rel) 81 03

USAGE (Wireless Radio LED) 09 C1

REPORT_SIZE (1) 75 01

OUTPUT (Data,Var,Rel) 91 02

REPORT_SIZE (7) 75 07

OUTPUT (Cnst,Var,Rel) 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 0A

COLLECTION (Application) A1 01

PHYSICAL_MINIMUM (0) 35 00

PHYSICAL_MAXIMUM (1) 45 01

LOGICAL_MINIMUM (0) 15 00

LOGICAL_MAXIMUM (1) 25 01

USAGE (Wireless Radio Slider Switch) 09 C2

REPORT_COUNT (1) 95 01

REPORT_SIZE (1) 75 01

INPUT (Data,Var,Abs) 81 02

REPORT_COUNT (1) 95 01

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 0A

COLLECTION (Application) A1 01

PHYSICAL_MINIMUM (0) 35 00

PHYSICAL_MAXIMUM (1) 45 01

LOGICAL_MINIMUM (0) 15 00

LOGICAL_MAXIMUM (1) 25 01

USAGE (Wireless Radio Slider Switch) 09 C2

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 C1

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 0A

COLLECTION (Application) A1 01

PHYSICAL_MINIMUM (0) 35 00

PHYSICAL_MAXIMUM (1) 45 01

LOGICAL_MINIMUM (0) 15 00

LOGICAL_MAXIMUM (1) 25 01

USAGE (Wireless Radio LED) 09 C1

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 use HID to reflect the state of the flight 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 has changed, the switch must consume this message and toggle the state.

Exceptions: Not Specified

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.

Scenarios:

User has a consistent predictable way to turn off all wireless capabilities on the PC

Success Metric: The certification tests will check that the appropriate HID message is sent and received, the radio management UI is updated to reflect the correct new state of the hardware button and when the user changes the state in the UI, a HID message must be sent to the LED and its state must be updated.

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.RadioManagement.RadioManagerCOMObject

Target Feature: System.Client.RadioManagement

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

    a. GPS

    b. Bluetooth radio that does not 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 will call 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 will call 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 ms. 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 ms.

Users do NOT need to have 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.

Note: If the system has an NFC radio, a radio manager will not be required. NFC will not appear in the radio management user interface.

Exceptions: Not Specified

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.

Scenarios:

For flight mode the user has a consistent predictable way to turn off all wireless capabilities on the PC. User has granular radio management. They can turn off/on individual radios to manage their power, privacy settings or to manage bandwidth usage. Also, on airplanes that support Wi-Fi, users can turn on Wi-Fi while still blocking other radios from broadcasting, and conserve battery power.

Success Metric: Pass/fail

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.RadioManagement.ConnectedStandby

Description:

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, Near Field Proximity, and Bluetooth radios that do not use the Windows native Bluetooth stack. Requirements in this section specifically apply to systems that are identified as supporting the Connected Standby feature.

Related Requirements:

- System.Client.RadioManagement.ConnectedStandby.NoRadioStatusIndicatorLights

System.Client.RadioManagement.ConnectedStandby.NoRadioStatusIndicatorLights

Target Feature: System.Client.RadioManagement.ConnectedStandby

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions: Not Specified

Business Justification:

The intent of a system that supports connected standby is that it is always connected to the cloud, whether the system is fully powered or in connected standby.

Scenarios:

Always connected

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments: Not Specified

System.Client.ScreenRotation

Description:

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

Target Feature: System.Client.ScreenRotation

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

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

Exceptions: Not Specified

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.

Scenarios:

User takes a new Windows 8 device and rotates the device from landscape to portrait mode

Due to the presence of the accelerometer Windows is able to automatically adjust the desktop screen to match the new orientation

There should be no cases where the driver does not support this rotation

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.Sensor

Description: Not Specified

Related Requirements:

- System.Client.Sensor.HumanProximitySensor
- System.Client.Sensor.Integrated
- System.Client.Sensor.MBBRequiresGPS

System.Client.Sensor.HumanProximitySensor

Target Feature: System.Client.Sensor

Title: System with human proximity sensor meets Windows requirements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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).

Exceptions: Not Specified

Business Justification:

This requirement ensures compatibility with Windows

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.Sensor.Integrated

Target Feature: System.Client.Sensor

Title: 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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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):

- 3D Accelerometer (SENSOR_TYPE_ACCELEROMETER_3D)
- 3D Inclinometer (SENSOR_TYPE_INCLINOMETER_3D)
- 3D Compass (SENSOR_TYPE_COMPASS_3D)
- 3D Gyrometer (SENSOR_TYPE_GYROMETER_3D)
- Device Orientation (SENSOR_TYPE_AGGREGATED_DEVICE_ORIENTATION)
- Ambient Light Sensor (SENSOR_TYPE_AMBIENT_LIGHT)

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

- 3D Compass
    - 3D accelerometer used for tilt compensation (required)
    - 3D Gyrometer used to enhance data rate and data integrity (required)
- 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)
- 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)

****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

In addition, each of these sensors should be correctly configured and calibrated with proper orientation (mounted in proper direction, etc.) as per the guidance specified for the specific sensor category as outlined in related whitepapers 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, please see the Sensors topic in the Device and Driver Technologies section of the Windows Driver Kit (WDK).

Exceptions: Not Specified

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.

Scenarios: Not Specified

Success Metric: Pass/Fail

Enforcement Date: Not Specified

Comments:

SYSFUND-8XXX

System.Client.Sensor.MBBRequiresGPS

Target Feature: System.Client.Sensor

Title: If a mobile broadband device is integrated into a tablet or convertible system, then an assisted GPS radio is required

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Assisted GPS is only required if a tablet or convertible includes a mobile broadband device.

Exceptions:

If the systems only solution for mobile broadband is WiMax, then this requirement does not apply.

Business Justification:

Assisted GPS is the preferred technology for mobile and outdoor location based scenarios including:

- Automotive and walking navigation applications.
- Show users position on a map.
- Find points of interest in the user's vicinity (local shops, restaurants etc.).
- Annotating content with location information (i.e. photos).
- Up-to-date local information (news, weather, traffic, etc.).
- Location-based social networking applications.
- Augmented reality applications (when combined with a digital compass and outward-facing camera).

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.SystemConfiguration

Description:

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

Target Feature: System.Client.SystemConfiguration

Title: System information

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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

Exceptions: Not Specified

Business Justification:

This information will be used by the Windows Certification team to review information about shipping systems versus systems that were submitted for certification.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.SystemConfiguration.Windows7NeccessaryDevices

Target Feature: System.Client.SystemConfiguration

Title: Windows 7 systems includes necessary devices

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client 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).

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.

Exceptions: Not Specified

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.

Scenarios:

Supports a wide range of scenarios that are offered by differing components. This also enables many Windows features such as the Desktop Windows Manager, and being able to playback audio.

Success Metric: Pass/Fail

Enforcement Date: June 1, 2007

Comments:

SYSFUND-0192

System.Client.SystemConfiguration.Windows8RequiredComponents

Target Feature: System.Client.SystemConfiguration

Title: Windows 8 systems must include certain devices

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Component requirements for Windows 8 tablets and convertibles can be found in requirements under System.Client.Tablet. For all other Windows 8 systems, the table down 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.

Any buttons that are implemented on the system cannot launch any customized user interface elements.

Component		Required
Storage	Space	At least 10GB of free space after completing OOBE
	Storage type	Meet minimum Microsoft Windows Storage requirements
System firmware		UEFI 2.3.1 as defined in System.Fundamentals.Firmware requirements
Networking	Ethernet or Wi-Fi	Must be either a lcertified Ethernet or Wi-Fi adapter
Graphics	GPU	Minimum of Direct3D 10 Feature Level 9_3 with WDDM 1.2 driver
	Video playback	Video playback of 1080p without glitches
	Minimum resolution	1024x768

Exceptions: Not Specified

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.

Scenarios:

These requirements satisfy Windows scenarios outlined in the rest of the Windows Certification Requirements.

Success Metric: Systems include required components.

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.SystemPartition

Description:

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

Target Feature: System.Client.SystemPartition

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

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client 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).**

Exceptions:

None

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.

Scenarios:

Enable Bitlocker scenarios and boot information.

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0032

System.Client.SystemPartition.OEMPartition

Target Feature: System.Client.SystemPartition

Title: Windows systems with recovery & OEM partitions must meet partitioning requirements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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).

Exceptions: Not Specified

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.

Scenarios:

BitLocker

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.Tablet

Description:

The requirements apply to systems that are a tablet or tablet/convertible system.

Related Requirements:

- System.Client.Tablet.BezelWidth
- System.Client.Tablet.ColdBootLatency
- System.Client.Tablet.RequiredHardwareButtons
- System.Client.Tablet.TabletPCRequiredComponents

System.Client.Tablet.BezelWidth

Target Feature: System.Client.Tablet

Title: Touch mobile systems must provide sufficient bezel space on the face of the device to allow it to be held without resulting in accidental touch input on the edges of the digitizer

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Touch-first mobile devices must provide sufficient bezel space on the face of the device to allow it to be held in the following postures without resulting in accidental touch input on the edges of the digitizer:

1. One or two hands gripping the device in landscape or portrait orientation, with thumbs/palms on the bezel
2. Device cradled in either landscape or portrait orientation, with fingers wrapping around the device on to the bezel
3. Device held from the bottom, with a thumb on the bezel

The bezel must not exceed 26 millimeters, to allow for users with smaller hands to access the edges of the digitizer while holding the device with two hands.

On tablet convertibles or similar systems, where there is a mechanical hinge, it is acceptable to allow for the bezel to exceed the 26 millimeter maximum up to 45 millimeters along the bottom edge closest to the hinge.

Example implementation: include a bezel of 20 millimeters on the face of the device, on all four sides of the digitizer/display, for a tablet device.

Exceptions: Not Specified

Business Justification:

The bezel width must provide sufficient space to all it to be held by the end user without resulting in accidental touch input.

Scenarios:

This is to support scenarios where there may be touchable elements around the edge of the screen.

Success Metric: Pass/Fail

Enforcement Date: Windows RC

Comments:

SYSFUND-8050

System.Client.Tablet.ColdBootLatency

Target Feature: System.Client.Tablet

Title: Time for I^2C touch controller to respond

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

The time from when power is applied to an I^2C touch controller to when the I^2C controller is responding to Human Interface Device (HID) commands and providing touch reports must not exceed 100 milliseconds.

Exceptions: Not Specified

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 I^2C touch controller be available to respond to HID commands and provide input reports almost immediately upon resume to ensure a consistent user experience.

Scenarios:

Windows Touch

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

New

System.Client.Tablet.RequiredHardwareButtons

Target Feature: System.Client.Tablet

Title: Tablet and Convertible PC must have five hardware buttons

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Hardware buttons must generate an event when a button is pressed. Any buttons that are implemented on the system cannot launch any customized user interface elements. The following buttons must be implemented on a tablet or convertible PC.

Button	Function
Power	Put the system into standby mode, press and hold for four seconds shuts down the system
Rotation Lock	Allows users to lock the display in a specific orientation
Windows Key Button	Navigation
Volume up and down button	Used to control the audio volume in the PC

The default orientation is in landscape mode and the Windows Key button must be on the front of the device facing the end user in the center along the bottom bezel. If the system is a convertible, the buttons must be accessible in all configurations. For convertible systems, it is acceptable to have the button off center along the bottom bezel when the convertible is in its tablet mode.

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

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

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 the ACPI specification (http://www.ACPI.info/spec.htm).

It is acceptable that the Power button be a slider switch as long as the switch automatically slides back to the normal state.

For Windows 8, the SAS signal will be sent when the combination of the Windows Key button and the Power Button is pressed.

The Windows button size is recommended to be at least 10.5 mm diameter in size and can be any shape (e.g. circular, square, rectangular).

The Windows button selects an item and resumes the device from the Connected Standby state.

The Windows button must be a soft press button type unless the front of the system is manufactured with optically bonded glass, then the buttons on the front of the bezel may be capacitive as long as they meet the following requirements:

1. The user can always feel the button. Some examples are but not limited to etching, raised bumps, etc.

2. The button always provides dynamic feedback. Some examples are but no limited to force feedback, haptic, button depresses and making clicking sound, etc.

3. Button sends signal on button down and button up. The capacitive button shall still be required to signal up and down via GPIO for SoC systems to be compatible with the inbox button driver.

4. The button must be able to recognize the difference between the Windows button press and edge swipe. If the end user happens to swipe in from the edge of the screen and happens to go over the button to invoke functionality of the application or the Windows 8 user experience (charms, snap or settings) from the edge, then the swipe gesture must work. The button must also be able to handle palm rejection in the scenarios where the end user is holding the system so their hand covers the button.

5. Touch controllers will be powered off when the screen is powered off so the capacitive button cannot utilize the same controller as the touch screen itself.

6. The capacitive button controller will need to remain powered in connected standby. If the Windows button is pressed while the system is in connected standby, it must wake the system. The system would still be subject to the same power draw requirement as documented in System.Fundamentals.PowerManagement.CS.ConnectedStandbyDuration.

Exceptions: Not Specified

Business Justification:

These will enable a navigation experience in the absence of a keyboard.

Scenarios:

The Windows start button is required for user navigation. In order to log onto the system in a domain environment, end users need to press Ctrl+Alt+Del. This needs to be sent in a control way so

pressing the combination of the Windows Key and the ACPI Power button is the method that will be used.

Success Metric: Pass/fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8005

System.Client.Tablet.TabletPCRequiredComponents

Target Feature: System.Client.Tablet

Title: Windows 8 Tablet and convertible PC must include certain devices

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

This version update includes the addition of Gyro, USBand device proximity to the list.

A **tablet form factor** is defined as a standalone device that combines the PC, display 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 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.

The battery is capable of charging without running Windows.

All components and devices used in these form factors must meet logo requirements and pass device logo testing for Windows 8 if the product type exists.

The minimum components that make up a tablet and tablet convertible are

Component		Required
Storage	**Space**	At least 10GB of free space after completing OOBE.
	Storage type	Meet minimum Microsoft Windows storage requirements
System firmware		UEFI firmware required as documented in section System.Fundamentals.Firmware
Networking	**WLAN**	Required
	Bluetooth 4.0 + LE	Required

Graphics	**GPU**	Minimum of Direct3D10 Feature Level9_3 with WDDM 1.2 driver
	Screen resolution	minimum 1366x768
Touch support		Logo d Windows 8 Touch solution
Camera	**Min Resolution**	720p. For complete requirements see section System.Client.Webcam
Ambient Light sensor	**Illuminance**	1-30k lux
	Dynamic range	5-60K
	ACPI	Minimum ACPI 3.0b compliant
Magnetometer		Required
Accelerometer sensor	**Axes**	3
	Data rates	>=50Hz
Audio output	**Speakers**	Required
Gyroscope		Required
USB 2.0		At least one USB2.0 controller and exposed port are required.

Exceptions: Not Specified

Business Justification:

Tablet and convertibles rely on multi-touch as the primary means of input. These systems are primarily focused on consumption scenarios such as web browsing, media, and casual gaming. It is likely that this form factor will also emerge in the enterprise as a productivity PC. These systems are optimized for consumption and light productivity. Requirements are based on these usage patterns.

Scenarios:

Requirements for components highlight scenarios from other section in the logo requirements.

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8004

System.Client.Tablet.Graphics

Description:

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

Target Feature: System.Client.Tablet.Graphics

Title: Tablet PCs support minimum resolution and color

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

The minimum native resolution/color depth is 1366x768 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 1366 (horizontally) and 768 (vertically).

Exceptions: Not Specified

Business Justification:

This is the minimum screen resolution for Tablet and Tablet convertible PCs.

Scenarios:

Minimum resolution requirements.

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8091

System.Client.Tablet.Graphics.SupportAllModeOrientations

Target Feature: System.Client.Tablet.Graphics

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

Applicable OS Versions:

- Windows 8 Client ARM
- Windows 8 Client x64
- Windows 8 Client x86

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.

Exceptions: Not Specified

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.

Scenarios:

- User takes a new Windows 8 device and rotates the device from landscape to portrait mode
- Due to the presence of the accelerometer Windows is able to automatically adjust the desktop screen to match the new orientation
- There should be no cases where the driver does not support this rotation
- Alternately, the user can also go to the Display Control Panel and rotate the desktop

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

New;

System.Client.UMPC.Graphics

Description:

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

Target Feature: System.Client.UMPC.Graphics

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

Applicable OS Versions:

- Windows 7 Client x64
- Windows 7 Client x86

Description:

For Windows 7, Ultra Mobile PCs are required to ship with at least Direct3D10 Feature Level 9_3and 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.

Exceptions: Not Specified

Business Justification: Not Specified

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

New; Win7 Graphics-0056 moved to system level requirement

System.Client.UserExperience

Description:

These requirements describe hardware requirements for the user experience.

Related Requirements:

- System.Client.UserExperience.PerfBenchMarks

System.Client.UserExperience.PerfBenchMarks

Target Feature: System.Client.UserExperience

Title: A system running Microsoft Windows meets required performance bench marks

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

End users expect a smooth and perform experience with their personal computer. A personal computer must achieve these performance metrics:

Action	Requirement
CPU working	2%
GPU working	5%
Glitches	Zero
Time to execute work load	200 milliseconds
Time to first frame	200 milliseconds
Frames dropped	0
Glitches	0

Scenarios to be tested include starting a process to showing the first frame, executing a search and moving windows on the screen.

Design Notes:

If the system has any of the following configurations:

- Multiple GPUs
- 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.

Exceptions: Not Specified

Business Justification:

Performance is critical to deliver a good end-user experience on Windows 8. The performance of the hardware subsystem and the graphics driver plays a big role in that.

Scenarios:

Scenarios to be tested include starting a process to showing the first frame, executing a search and moving windows on the screen.

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.VideoPlayback

Description:

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.GlitchfreeHDVideoPlayback

Target Feature: System.Client.VideoPlayback

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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 24-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 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) 5mbps video stream single frame decode + DXVA decode device creation time + DX surface/texture allocation time + frame decode < 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 GPUs
- 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.

Exceptions: Not Specified

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.

Scenarios:

The system is able to playback high definition video without any glitches.

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

New; Win7 SYSFUND-0062

System.Client.VideoPlayback.GlitchfreePlayback

Target Feature: System.Client.VideoPlayback

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

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client 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

Exceptions: Not Specified

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 users expect.

Scenarios:

The ability of the system to playback high definition video without any glitches.

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-0062

System.Client.Webcam

Description:

These requirements apply to cameras integrated into the system.

Related Requirements:

- System.Client.Webcam.PhysicalLocation
- System.Client.Webcam.VideoCaptureAndCamera

System.Client.Webcam.PhysicalLocation

Target Feature: System.Client.Webcam

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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 and should match the present display orientation of landscape or portrait. 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.

All other fields in the _PLD are optional.

For more information see the ACPI specification section 9.13.

Exceptions: Not Specified

Business Justification:

This enables developers the ability to write applications that take advantage of front or rear facing cameras.

Scenarios:

This enables a common developer platform.

Success Metric: Pass/Fail

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.Webcam.VideoCaptureAndCamera

Target Feature: System.Client.Webcam

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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: http://msdn.microsoft.com/en-us/library/ff559750(v=VS.85).aspx,with correct AcpiUid. Later IHV driver should report the correct connect status in DxgkDdiQueryChildStatus: http://msdn.microsoft.com/en-us/library/ff559754(v=VS.85).aspx.)

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.

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). 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.

Resolution & Performance

At least 720p 30 fps must be supported by the capture pin.

Controls

Following controls are recommended:

- Brightness
- Contrast
- Hue
- White Balance
- Backlight Compensation
- Zoom
- Exposure
- Focus

Image Stabilization

If there is image stabilization on the device, device must use the Microsoft defined GUID ("1A651206-EB69-4E83-B38F-7DC4CB21442C") for turning on/off this functionality.

If available, latency due to Image Stabilization shall not be more than one frame buffering.

Mirroring

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

Exceptions: Not Specified

Business Justification:

Developers will have the ability to write applications that can use any integrated webcams.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.Webcam.Specification

Description:

Video capture devices are cameras and audio/video input devices that bring streaming data into a PC. Cameras that are 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

Target Feature: System.Client.Webcam.Specification

Title: Video capture and cameras must support resolution, anti-flicker, and audio-video sync

Applicable OS Versions:

- Windows 8 Client ARM
- Windows 8 Client x64
- Windows 8 Client x86

Description:

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

Feature	Specification
Video Resolution, frame rate	>=720P (Required to support specific resolutions of 1280x720 and640x480): =30 FPS @ 200 lux
Image resolution	>=0.92 MP (Required to support the resolution of 1280x720)
Anti-Flicker Solution	Manual Support
Audio-Video sync	Audio leads video <= 45 ms, Audio lags video <= 90 ms

Exceptions: Not Specified

Business Justification:

The camera needs to support scenarios such as taking a photo, recording video or real-time communication.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Client.WindowsMediaCenter

Description:

These requirements are describing the requirements for TV tuners and remotes for Windows Media Center.

Related Requirements:

- System.Client.WindowsMediaCenter.WMCRemote

System.Client.WindowsMediaCenter.WMCRemote

Target Feature: System.Client.WindowsMediaCenter

Title: If including a remote control and receiver with a system it must comply with the Windows Media Center Remote Control and Receiver/Transceiver Specification document

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64

Description:

A Windows Media Center remote control is optional. If any remote is shipped with a system with Windows Media Center, then it must comply with the Windows Media Center Remote Control and Receiver/Transceiver Specifications.

All Windows Media Center remote controls must include the Windows Media Center Green Button Design as specified in the Windows Media Center Remote Control and Receiver/Transceiver Specification and associated artwork.

If a system includes only an analog TV tuner card(s) capable of receiving a signal from a TV Set Top Box and ships a remote control with matching receiver/transceiver, the system must have the ability to control a Set Top Box via IR. IR learning and IR emitting functions are required.

If shipping a system with digital tuners, IR learning and IR emitting is optional. If shipping a laptop system, IR learning and IR emitting is optional.

Design Notes:

Microsoft recommends that the remote controls use either MC RC 6 IR protocol or the WMC QP IR protocol.

Exceptions: Not Specified

Business Justification:

Consistent end user experience when using Windows Media Center.

Scenarios:

Compatibility with Windows Media Center.

Success Metric: Pass/Fail

Enforcement Date: Windows 7 RC

Comments:

SYSFUND-0042

System.Fundamentals.DebugPort

Description:

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

Target Feature: System.Fundamentals.DebugPort

Title: System exposes debug interface that complies with Debug Port Specification

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64
- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 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 in order of preference:

- Ethernet Network Interface Card from the supported list: http://go.microsoft.com/fwlink/?LinkId=237141
- 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). See the supported hardware list here: http://go.microsoft.com/fwlink/?LinkId=237141
- 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 REQUIREMENTS APPLY:

- There must be at least one user accessible debug port on the machine.
- 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=58376

Exceptions: Not Specified

Business Justification:

This requirement is needed to ensure that all hardware solutions running Windows are debug-able.

Scenarios:

This is to support the ability to connect a kernel debugger to a PC or device

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0003

System.Fundamentals.DebugPort.USB

Description:

The ability to debug a USB 3 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 sport.

Related Requirements:

- System.Fundamentals.DebugPort.USB.SystemExposesDebugInterfaceUsb

System.Fundamentals.DebugPort.USB.SystemExposesDebugInterfaceUsb

Target Feature: System.Fundamentals.DebugPort.USB

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64

Description:

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

- Per the XHCI debug specification, all USB 3 XHCI controller ports must support debugging.
- USB 3.0 hubs must not be integrated into the chipset. All USB3 ports exposed by a system chipset must support XHCI debug.
- Ideally all user accessible USB 3.0 ports on a machine should be debug capable. If a machine has both debug capable USB 3 ports as well as USB 3 ports that do not support debugging, the debug capable ports must be identifiable by either permanently marked so that they can be distinguished from the ports that do not support debugging, including it in the help documentation or from a support website.

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

Exceptions: Not Specified

Business Justification:

The goal of this requirement is to ensure systems are debug-capable in the scenario in which USB 3 devices are being used.

Scenarios:

This is to support the ability to connect a kernel debugger to a PC.

Success Metric: Pass/Fail

Enforcement Date: 3/24/2011

Comments:

SYSFUND-8006

System.Fundamentals.Firmware

Description:

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.InternalPointingDeviceWorksWithExternalPointingDevice
- System.Fundamentals.Firmware.UEFIBitLocker
- System.Fundamentals.Firmware.UEFIBootEntries
- System.Fundamentals.Firmware.UEFICompatibility
- System.Fundamentals.Firmware.UEFIDefaultBoot
- System.Fundamentals.Firmware.UEFIEncryptedHDD
- System.Fundamentals.Firmware.UEFILegacyFallback
- System.Fundamentals.Firmware.UEFIPostTime
- System.Fundamentals.Firmware.UEFISecureBoot
- System.Fundamentals.Firmware.UEFITimingClass

System.Fundamentals.Firmware.ACPI

Target Feature: System.Fundamentals.Firmware

Title: ACPI System Requirements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

ACPI 5.0 is the relevant revision of the Spec for these requirements.

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
Firmware Performance Data Table (FPDT)	Required

DSDT Requirements

As per ACPI 5.0, 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 a 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 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. In addition, systems that support connected standby must:

- Implement the ACPI Control Method Power Button using a dedicated GPIO interrupt pin to signal button press events.
- Configure the power buttons GPIO interrupt pin as a non-shared, wake-capable (ExclusiveAndWake) GPIO interrupt connection resource.
- List the Power Buttons 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 buttons 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 buttons GPIO Interrupt connection resource first in the Button Array devices _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 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 devices Time capabilities.

Exceptions: Not Specified

Business Justification:

Devices and buses that are on the system must have an ACPI namespace to ensure compatibility with Windows.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.Firmware.ACPIRequired

Target Feature: System.Fundamentals.Firmware

Title: System that support connected standby must contain required ACPI elements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

ACPI 5.0 is the relevant revision of the Spec for these requirements.

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

Exceptions: Not Specified

Business Justification:

Devices and buses that are on the system must have an ACPI namespace to ensure compatibility with Windows.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.Firmware.FirmwareSupportsBootingFromDVDDevice

Target Feature: System.Fundamentals.Firmware

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

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

Systems are required to comply with this requirement in order to boot from a DVD drive.

Scenarios:

Boot the system from DVD device

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0084

System.Fundamentals.Firmware.FirmwareSupportsUSBDevices

Target Feature: System.Fundamentals.Firmware

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

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server 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.

Design Notes:

Microsoft recommends that the system firmware support hot plug of input devices after system firmware has started execution. OEMs are encouraged to test the boot functionality by creating a bootable USB flash drive with WinPE. See the OEM Pre-installation Kit (OPK) for details. Vendors may license WinPE (at no charge). For information, send e-mail to licwinpe@microsoft.com. To indicate that an USB end-point device is internal, it is strongly recommended that "Internal Device" be added to the end of the name string so that it shows up as an internal device in Device Manager. If the device uses an inbox INF a second INF can be created (and signed by Microsoft using the submission process) with the additional text string that uses Needs / Includes to call the inbox INF. See INF in the Windows Driver Kit.

Exceptions:

None

Business Justification:

Without this requirement, users could purchase a system and try to install Windows on it and not be able to because they cannot use their keyboard or mouse.

Scenarios:

Change settings pre-boot environment with mouse and keyboard.

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0071

System.Fundamentals.Firmware.InternalPointingDeviceWorksWithExternalPointingDevice

Target Feature: System.Fundamentals.Firmware

Title: System that has an integrated or internal pointing device either disables the device or enables dual operation if external pointing device is attached

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

If dual operation is not possible, the system firmware must disable the internal pointing device by default when any external pointing device is detected. When the internal pointing device is disabled in system firmware, no input from the embedded pointing device must be sent to the operating system.

Exceptions:

None.

Business Justification:

This is for when an external mouse is plugged into a notebook, the system should either disable the internal mouse or touchpad, or both should function properly.

Scenarios:

This is for when an external mouse is plugged into a PC with an internal mouse.

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0026

System.Fundamentals.Firmware.UEFIBitLocker

Target Feature: System.Fundamentals.Firmware

Title: 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)

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64
- Windows Server 2008 Release 2 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

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 Certified system. UEFI is the core prerequisite of the Windows 8 platform that enables key features in security and performance.

Scenarios:

Reliability and compatibility with BitLocker

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8107 BitLocker can still be enabled on systems that don't have wired network support, and thus do not have EFI_DHCP{4|6}_PROTOCOL, but Network Key Protector will not be available in this case.

System.Fundamentals.Firmware.UEFIBootEntries

Target Feature: System.Fundamentals.Firmware

Title: UEFI firmware honors software control over load option variables

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64
- Windows Server 2008 Release 2 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 boot manager 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

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).

Scenarios:

Boot and run software from USB

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8105

System.Fundamentals.Firmware.UEFICompatibility

Target Feature: System.Fundamentals.Firmware

Title: System firmware must meet Windows Compatibility requirements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Other requirements may add additional sections of compatibility to this list, but this is the baseline.

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

Exceptions:

None.

Business Justification:

Compatibility with Windows.

Scenarios:

Compatibility

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

X64-capable systems must boot into Windows 64 bit UEFI mode by default. X64 capable systems must not boot Windows 32 bit UEFI mode. If a system is X86-capable only and supports Connected Standby, the system must boot into Windows 32 bit UEFI mode by default. All UEFI mode systems must have GUID Partition Table (GPT) formatted boot disks.

System.Fundamentals.Firmware.UEFIDefaultBoot

Target Feature: System.Fundamentals.Firmware

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64
- Windows Server 2008 Release 2 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 and applies only if a Server system is UEFI capable

Exceptions:

None. All versions of Windows 8 shall be UEFI-compatible.

Business Justification:

Support of UEFI is a critical element of the overall Windows 8 experience.

Scenarios:

Secured Boot

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

Although a system may ship with a non-UEFI-compatible OS such as Windows 7, should such a system ever be upgraded to a UEFI-compatible OS such as Windows 8, noncompliance with this requirement will result in some features not working.

System.Fundamentals.Firmware.UEFIEncryptedHDD

Target Feature: System.Fundamentals.Firmware

Title: 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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server 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 Encrypted Drive 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

Exceptions: Not Specified

Business Justification:

Support of UEFI is a critical element of the overall Windows 8 experience.

Scenarios:

BitLocker

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8103

System.Fundamentals.Firmware.UEFILegacyFallback

Target Feature: System.Fundamentals.Firmware

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64
- Windows Server 2008 Release 2 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 ROMs 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

Exceptions:

None. All versions of Windows 8 shall be UEFI-compatible.

Business Justification:

Support of UEFI is a critical element of the overall Windows 8 experience.

Scenarios:

Secured Boot

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

Although a system may ship with a non-UEFI-compatible OS, should such a system ever be upgraded to a UEFI-compatible OS, noncompliance with this requirement will result in some features not working.

System.Fundamentals.Firmware.UEFIPostTime

Target Feature: System.Fundamentals.Firmware

Title: System must meet POST time requirements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64

Description:

All systems meet the following S4 POST time requirements as reported through an ACPI table. POST begins with system firmware initialization and ends with hand off to the OS Loader. POST is measured from the S4 state because "hybrid boot" is the default boot path in Windows 8 and starts from the S4 system state as defined in ACPI.

- Systems with SSD or hybrid SSD must POST in 2 seconds or less. A hybrid SSD has both SSD storage and spinning disks.
- Systems with 2.5 inch spinning disks must POST in 4 seconds or less.
- If a system has a TPM and initializes it, an additional 300ms can be added to minimum boot time requirements. (E.g., 2.3 seconds for SSD based systems.)

Exceptions:

The following configurations can be exempted from this logo requirement:

- Systems with RAID implementations
- Systems that attach to a non SATA bus (E.g., SCSI or Fibre Channel)
- Systems with disks that spin a rates of 10,000 RPM or greater
- Systems with ECC or parity memory
- Systems with boot disks that have 3 or more platters

- Systems with 3.5" boot disks

Business Justification:

Increase system performance.

Scenarios:

Performance

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8114

System.Fundamentals.Firmware.UEFISecureBoot

Target Feature: System.Fundamentals.Firmware

Title: All client systems must support UEFI Secure boot

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64
- Windows Server 2008 Release 2 x64

Description:

Note: These requirements are If Implemented for Server systems and applies only if a Server system is UEFI capable.

1. *MANDATORY: 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 A,* and UEFI SWG approved change requests ECR-847, ECR-848, ECR-855, ECR-858, ECR-866*:*

   7.1, 7.2, 7.2.1, 27.2, 27.5 through 27.8 (as further profiled below)

2. *MANDATORY.* ***Secure Boot must ship enabled*** *(i.e., UEFI Version 2.3.1 Errata A variables SecureBoot=1 and SetupMode=0) and with a signature database (EFI_IMAGE_SECURITY_DATABASE) necessary to boot the machine securely pre-provisioned. 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". See also bullet 12, below.*

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".

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

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

6. *MANDATORY. 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. See also bullet 14.*

7. *MANDATORY:* ***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. *MANDATORY:* ***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 rollback, including rolling back to older or 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. 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. *MANDATORY:* ***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 execute. 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. See also bullet 13.

10. *MANDATORY**. 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.*

11. *MANDATORY: **Protection required for Non-Volatile Firmware Storage after ExitBootServices().** After ExitBootServices() is called, hardware protection shall enforce the behaviors defined in the UEFI specification Version 2.3.1 Errata A, section 7.2, for access to non-volatile firmware storage. After ExitBootServices() is called, only variables that have the EFI_VARIABLE_RUNTIME_ACCESS attribute set may be modified, and any variables that lack the EFI_VARIABLE_RUNTIME_ACCESS attribute must not be accessible. Furthermore, hardware protection shall prevent the OS from unauthenticated access to authenticated variables, as well as any access to boot services variables. For example, malicious code executing in the context of the operating system must not be able to circumvent UEFI Runtime Services by any mechanism; for example, direct communication with flash hardware.*

12. *MANDATORY: **The UEFI firmware shall include a MS-provided 2048-bit public RSA key in the form of an X.509 certificate in the allowed signature database.** All Win8 client platforms must support UEFI Secure Boot (as defined in UEFI v2.3.1 chapter 27) and must ship with Secure Boot enabled by default and include a PK that is set and a valid KEK database as well as a valid signature database (EFI_IMAGE_SIGNATURE_DATABASE). The system will use this database to verify that only trusted code(e.g. trusted signed boot loader) is initialized, and that any unsigned image or an image that is signed by an unauthorized publisher will not execute. The contents of the signature database is determined by the OEM, based on the required native and 3rd party UEFI drivers, respective recovery needs, and the OS Boot Loader installed on the machine. A Microsoft-provided signature, 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.*

13. *MANDATORY**. All firmware components shall be signed using RSA-2048 with SHA-256** (this is the default policy for acceptable signature algorithms).*

14. *MANDATORY: **The UEFI firmware identifies a forbidden signature via the forbidden signature database** (EFI_IMAGE_SIGNATURE_DATABASE1). 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.*

15. *MANDATORY: **Microsoft Key Encryption Key (KEK) is provisioned** A valid Microsoft-provided KEK shall be included in the KEK database. Microsoft will provide the KEK in the form of either an EFI_CERT_X509_GUID or EFI_CERT_RSA2048_GUID type signature. The Microsoft KEK*

signature shall use the following SignatureOwner GUID: {77fa9abd-0359-4d32-bd60-28f4e78f784b}.

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

17. *MANDATORY:* ***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.*

18. *MANDATORY:* ***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. Execution of these tools and shells must require that a platform administrator disables Secure Boot.*

19. *MANDATORY:* ***Secure Boot Variable.*** *The firmware shall implement the Secure Boot variable as documented in Section 3.2 "Globally Defined Variables' of UEFI Specification Version 2.3.1 Errata A"*

20. *MANDATORY: 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.

b) If the user ends up deleting the PK then, upon exiting the Custom Mode firmware setup, the system will be operating in Setup Mode with Secure Boot 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 enable.

21. *MANDATORY:* ***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. Programmatic disabling of Secure Boot either during Boot Services or after exiting EFI Boot Services MUST NOT be possible. Disabling Secure MUST NOT be possible on ARM systems.*

22. *MANDATORY:* ***If the firmware is reset to factory, 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 PK_{pub} shall be re-established along with the original, manufacturer-provisioned signature databases.*

23. *MANDATORY:* ***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.*

24. *MANDATORY:* ***A working WinRE image must be present on all Win8 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 WinRE image must be present and enabled by default. As part of the Windows 8 Trusted Boot work enhancements have been made to WinRE to allow optimized recovery from signature verification failures in Secure Boot. OEMs must include WinRE as part of their factory image on all Win8 client systems.*

25. *MANDATORY:* ***Firmware-based backup and restore****. If the OEM provides a mechanism to backup boot critical files (e.g. 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.*

26. *MANDATORY:* ***Firmware-based backup synchronization****. Backup copies of boot critical components (e.g. 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, 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 a Windows Update patch, then the backup copy of bootmgr.efi stored in firmware must be updated on the next boot.*

27. *MANDATORY:* ***All Win8 client systems must support a secondary boot path****. For all Win8 systems configured for Secure Boot, there must be an alternate boot path option that will be followed by the firmware in the event that the primary Windows OS loader fails. The second boot path will 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.*

28. *MANDATORY: **All Win8 client systems must support a USB boot path for recovery purposes**. For all Win8 systems configured for Secure Boot, there will be a last resort of booting from USB.*

29. *MANDATORY. **Supporting GetVariable() for the EFI_IMAGE_SECURITY_DATABASE** (both authorized and forbidden signature database) and the Secure Boot variable.*

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

31. *MANDATORY: **Reserved Memory for Windows Secure Boot UEFI Variables**. There is no maximum NVRAM storage size limit. However, for all Win8 systems, a total of 64kB of non-volatile NVRAM storage memory must be reserved for Secure Boot UEFI variables (authenticated and unauthenticated) used by Windows, and the maximum variable size allowed is 32kB. The total space used by existing Windows Secure Boot variables plus the remaining free space must be equal to or greater than 64kB. {64kB = Free Space + sizeof(PK + KEK + db + dbx + all variables in Microsoft namespace)}*

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

    - *The Secure Boot state & configuration (PK, KEK, db, dbx, SetupMode, SecureBoot)*
    - *All UEFI variables with VendorGuid {77fa9abd-0359-4d32-bd60-28f4e78f784b}*
    - *A physically-present user who authenticates to the firmware may change, reset, or delete these values*

32. *MANDATORY. **The platform shall support EFI variables** that are:*

- *accessible only during the boot services or also accessible in the runtime phase;*
- *non-volatile; and*
- *possible to update only after proper authorization (i.e. properly signed).*

I.e., the platform must support EFI variables with any valid combination of the following UEFI 2.3.1 variable attributes set:

EFI_VARIABLE_NON_VOLATILE
EFI_VARIABLE_BOOTSERVICE_ACCESS

EFI_VARIABLE_RUNTIME_ACCESS
EFI_VARIABLE_AUTHENTICATED_WRITE_ACCESS

EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS

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

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

Exceptions:

None

Business Justification:

Support of UEFI is a critical element of the overall Windows 8 experience.

Scenarios:

Secure Boot

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

Although a system may ship with a non-UEFI-compatible OS, should such a system ever be upgraded to a UEFI-compatible OS, noncompliance with this requirement will result in some features not working.

System.Fundamentals.Firmware.UEFITimingClass

Target Feature: System.Fundamentals.Firmware

Title: System firmware must expose timing and class information

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

System firmware shall declare its runtime performance measurements by conforming to ACPI 5.2.23 with a populated FPDT. Specifically, the firmware basic boot performance data record defined in 5.2.23.7 and 5.2.23.8 shall be populated.

ACPI 5.2.23 is equally valid for Class 2 and Class 3 UEFI boots as well as Class 2 boots with CSM (for example, see Table 5-109). During a Class 2 boot with CSM, several fields in the performance record will be set to zero, indicating a non-UEFI boot (and thus a Class 2 system firmware implementation). No additional class information is required to meet the requirement.

Exceptions:

None. All Windows 8 versions shall be ACPI 5.0 capable.

Business Justification:

All of Windows is instrumented for co-engineering collaboration and as an aid to debugging and tuning. Instrumentation is also good for detecting component interactions and regressions. System firmware is a notable exception to this. This requirement will close that gap.

Scenarios:

Reliability and co-engineering.

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

ACPI 5.2.23 also includes tables for declaring S3 suspend and S3 resume performance. It is recommended that firmware also update these fields but it is not required. The exact method for the system firmware to take the measurement of its own performance is not defined here. Each firmware implementation may define its own method so long as it conforms to the ACPI 5 spec.

System.Fundamentals.Firmware.Boot

Description:

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

Target Feature: System.Fundamentals.Firmware.Boot

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

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client 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.

Exceptions: Not Specified

Business Justification: Not Specified

Scenarios: Not Specified

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0080,

System.Fundamentals.Firmware.Boot.SystemWithBootDeviceGreaterThan

Target Feature: System.Fundamentals.Firmware.Boot

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 7 Client x64
- Windows 7 Client x86
- Windows Server 2008 Release 2 x64
- Windows 8 Server x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

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.

Scenarios:

Device compatibility

Success Metric: Pass/Fail

Enforcement Date: December 1, 2010

Comments:

SYSFUND-0229

System.Fundamentals.Firmware.CS

Description:

This feature includes requirements specific to system firmware for systems that support connected standby.

Related Requirements:

- System.Fundamentals.Firmware.CS.CryptoCapabilities
- System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby

System.Fundamentals.Firmware.CS.CryptoCapabilities

Target Feature: System.Fundamentals.Firmware.CS

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client ARM
- Windows 8 Client x64

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. **MANDATORY**. 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:

    1. 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.

    2. 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.

    3. 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. **MANDATORY**. The platform must implement the EFI_HASH_PROTOCOL from UEFI Industry Group, Unified Extensible Firmware Interface Specification version 2.3.1 Errata A Section 27.4, UEFI SWG ECR-850 and ECR-853. 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 protocol's EFI_HASH_ALGORITHM_SHA256_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. **MANDATORY.** 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. **MANDATORY.** 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. **Mandatory**. 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 encryption and decryption as measured at the CNG kernel-mode BCrypt interface when processing 8 kByte blocks, and >=40 MBytes/s when processing 4kByte 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	M			Performance >60 Mbytes/s as measured at the CNG kernel-mode BCrypt interface when processing 4kByte blocks.
HMAC-SHA1	O			
SHA-256	M			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]TheConnected 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.

Exceptions: Not Specified

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.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

SYSFUND-8312

System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby

Target Feature: System.Fundamentals.Firmware.CS

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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

1. **MANDATORY.** Connected-Standby systems must meet all of the requirements cited in this section and under "system.fundamentals.firmware.uefisecureboot"section.

2. **MANDATORY**. **Boot Integrity.** SoC hardware will use 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 will provide power-on reset logic to execute from on-die ROM or secure on-die SRAM.

3. **MANDATORY.** 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 (i.e.: 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.

4. **MANDATORY.** 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.

5. **MANDATORY.** Upon power-on, the platform shall start executing boot firmware stored on-die (i.e. read-only) 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.

6. **MANDATORY.** Protection of physical memory from unauthorized internal DMA (e.g. 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.

7. **OPTIONAL**. The memory containing the initial boot firmware (executing in SRAM) must be made inaccessible upon jumping to the next validated stage of the boot sequence and the only way for the system to access this code again is to perform a power-on-reset.

8. **MANDATORY.** The platform shall enforce set policy with regards to 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 component cannot, without proper authorization (e.g. physical presence), be replaced by an earlier version component where earlier and later may be defined by a (signed) version number, for example.

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

10. **MANDATORY.** Physical Security Requirements. In retail parts, once the SoC 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, disable all hardware test modes disable all hardware scan chains.

12. **MANDATORY.** On ARM platforms Secure Boot Custom Mode is not allowed. A physically present user cannot override Secure Boot authenticated variables (i.e. PK, KEK, db, dbx).

13. **MANDATORY**. 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.

Exceptions: Not Specified

Business Justification:

Systems that support connected standby need to be secure.

Scenarios:

Secured Boot

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

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

System.Fundamentals.Graphics

Description:

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.Windows7.MultipleOperatingModes

System.Fundamentals.Graphics.FirmwareSupportsLargeAperture

Target Feature: System.Fundamentals.Graphics

Title: 32-bit and 64-bit system firmware supports large aperture graphic adapters

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server 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.

Exceptions: Not Specified

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.

Scenarios:

Graphics adapter just works.

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

This was the Win7 SYSFUND-0073 requirement

System.Fundamentals.Graphics.MicrosoftBasicDisplayDriver

Target Feature: System.Fundamentals.Graphics

Title: System is compatible with the Microsoft Basic Display Driver

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

- Windows 8 Server 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.

Exceptions: Not Specified

Business Justification:

Ensures compatibility with the Windows OS.

Scenarios:

Driver reliability and compatibility.

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

New; SYSFUND-8090

System.Fundamentals.Graphics.MultipleOperatingMode

Target Feature: System.Fundamentals.Graphics

Title: If a Windows 8 system has Multiple GPU's, the graphics and system test must pass in every "Operating Mode"

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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 feature, 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 GPUs increases the complexity of the overall system and may expose the system to Windows 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 a contingency to certify these features do not have explicit support by Microsoft in the OS runtimes, APIs 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 via contingency 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 a contingency (type 2 above) must carry the responsibility for testing, supporting and servicing any divergence from supported features. To be certified, the devices additional quality assurance and future support will be documented using the contingency process.

For an IHV to ship a type 2 certified product the IHV will request a contingency that:

1. Declares the intent to ship a feature that is not supported by Microsoft Windows

2. Provides Microsoft with a detailed test plan to exercise the feature and assure no Windows features fail as a result of the feature.

3. Upon approval of the test plan, evidence of a passing submission of the device, and documentation, the contingency may be extended to the device.

For an OEM to certify a system that includes a type 2 device, the OEM will request a contingency which references the agreement between the IHV and Microsoft.

The conditions of the contingency are:

1. The OEM shall pass the system integration tests documented by the device manufacturer and approved by Microsoft.

2. OEM and IHV shall monitor the reliability of the Windows unsupported feature using telemetry.

3. OEM and IHV shall correct any end-user issues resulting in Windows feature, application or kernel failures observed for a period of 5 years after certifying a system with the device/driver, and supporting the device on the next operating system for a period of 2 years, whichever is period is longer.

4. OEM and IHV shall distribute any updates required to meet the support described in #3 via WU to all systems that include the feature, in a timely manner. Timely manner is defined as:

    a. Updates will be provided no more than 3 months after a failure is discovered.

    b. Updates will be provided at least 3 months before a patch, service pack or future OS.

5. Microsoft, in collaboration with the IHV, may continue to update and distribute a driver for the device beyond the contingency 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 contingency 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 feature 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 APIs 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 a contingency.

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 a contingency 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 contingency
Linked Adapter	Not Allowed	Windows Supported
i- GPU with a Secondary d-GPU	Not Allowed	Requires contingency
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. 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 GPUs. Examples are AMD Crossfire 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 and AMD Hybrid CrossFire X.

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.

Exceptions: Not Specified

Business Justification: Not Specified

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.Graphics.NoRebootUpgrade

Target Feature: System.Fundamentals.Graphics

Title: Graphics drivers must be upgradable without a reboot of the system

Applicable OS Versions:

- Windows 8 Server x64
- Windows 8 Client ARM
- Windows 8 Client x64
- Windows 8 Client x86

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.

Exceptions: Not Specified

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.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

New; Sysfund-8500

System.Fundamentals.Graphics.Windows7.MultipleOperatingModes

Target Feature: System.Fundamentals.Graphics

Title: If a Windows 7 system has Multiple GPU's, the graphics and system test must pass in every "Operating Mode"

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client 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 GPUs. 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.

Exceptions: Not Specified

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.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.Graphics.Display

Description:

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

Target Feature: System.Fundamentals.Graphics.Display

Title: System minimum resolution and color depth

Applicable OS Versions:

- Windows 7 Client x64
- Windows 7 Client x86
- Windows Server 2008 Release 2 x64

Description:

Minimum resolution/color depth is 1024x768 at a depth of 32bits on each output simultaneously.

Exceptions:

Business Justification:

These are minimum resolution requirements for non-tablet PC systems.

Scenarios:

Minimum resolution

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

System.Fundamentals.Graphics.DisplayRender

Description:

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.DWMSystemPerformance
- System.Fundamentals.Graphics.DisplayRender.Performance
- System.Fundamentals.Graphics.DisplayRender.StableAndFunctional

System.Fundamentals.Graphics.DisplayRender.DWMSystemPerformance

Target Feature: System.Fundamentals.Graphics.DisplayRender

Title: A Windows 8 system has sufficient performance to provide a good user experience for mainstream composition and animation scenarios on both A/C and battery power

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Users expect a smooth and responsive experience with their mainstream applications. As such, good graphics performance is crucial for all Windows 8 systems.

All Windows 8 systems must have sufficient graphics performance to meet performance requirements and metrics for a collection of scenarios that represent mainstream desktop composition and window management scenarios.

Since users typically have multiple applications running at the same time, it is important that the cost of typical user interactions with the application windows should be low. In addition, since

applications will regularly update the client area of windows when rendering new content, the cost of updating the client area should be low.

The tests will use a number of mini-apps to simulate basic windowing scenarios. These mini-apps evaluate overall system performance using a number of common metrics.

Metrics

The following are the metrics that will be measured for the applicable scenarios and used to determine the performance of a system.

CPU time per frame

The average CPU time taken by DWM.exe to compose each frame.

GPU time per frame

The average GPU time taken by DWM.exe to compose each frame.

Video Glitches

The standard deviation of frame period for the duration and frame rate of the scenario are used to measure glitches.

Composition latency

Elapsed time between frame start and frame end; frame end is the time when the frame processing is completed on the GPU).

Design notes:

These tests will only run on battery power if the system is a laptop or tablet/convertible. If the system is a desktop, the tests will be run on A/C only.

The following scenarios will be instrumented and will have goals for the metrics described:

mini-applications

Scenario	Metrics			
	CPU Time per Frame	GPU Time per Frame	Glitching	Composition latency
Simple App updating the client area **Open one window which is updating the client area of the window.**	Four milliseconds	Four milliseconds	Zero glitches at 60 frames per second	16 milliseconds
Move a Window **Open a window and move window in linear fashion.**				
Drag a Window over other windows **Open some windows on top of each other; then open another window and move the new window over other opened windows**				

Minimize - Restore **Minimize a Window and then restore it. The minimize/restore occurs with an animation. Ensure that the animation is smooth**				

Design Notes:

If the system has any of the following configurations:

- Multiple GPUs
- 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.

Exceptions: Not Specified

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.

Scenarios:

Great end user experience.

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.Graphics.DisplayRender.Performance

Target Feature: System.Fundamentals.Graphics.DisplayRender

Title: 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

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64

Description:

Users expect a smooth and responsive experience with their mainstream applications. As such, good graphics performance is crucial for all Windows 8 systems.

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 will encompass both high-level scenarios with applications like the Internet Explorer, including CSS animations and Windows Shell and DirectX mini-scenarios.

The high-level scenarios represent end-user perceived performance for commonly used applications.

The DirectX scenarios will be 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 will be 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 will be 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 is meant to measure the total memory cost of graphics for a scenario. To calculate it, several different measurements are added:

- User-mode reference set: all of the user-mode system memory pages touched by the application process during the test.
- Driver and dxgkrnl memory: kernel-mode system memory allocated by dxgkrnl and the KMD during the test. This includes:
    - Paged and nonpaged pool (pool is like a heap shared by all kernel-mode code)
    - Locked pages (non-pageable memory allocated by the page)
    - Contiguous pages (non-pageable physically contiguous memory)
    - Pages for MDL (memory allocated for page lists)
- Video memory

Glitching

For scenarios with animations, users expect smooth animations at 60 Hz. This metric looks for the total # of frames missed and # 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 will also verify that DWM glitches are zero.

Design notes:

These tests will only run on battery power if the system is a laptop or tablet/convertible. If the system is a desktop, the tests will be run on A/C only.

The following scenarios will be instrumented and will 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 15 milliseconds or less for this test. The UMD and KMD creation times are part of the time to first frame measurements.	NA	NA
Direct2D BrushTest **Draw a bitmap with the following brush types:** **1. Bitmap brush with different extend modes (clamp, wrap and mirror)** **2. Solid color brush** **3. Gradient brush** **a. Linear gradient** **b. Radial gradient** **4. Image Brush**	200 milliseconds	NA	60 FPS
Direct2D Charts **rendering charts, similar to Office Excel, Word and Visio. Most of these charts use basic geometry primitives like lines, rectangles, circles etc. and small text strings.**	200 milliseconds	No glitching at 60 fps	60 FPS
HTML5 Canvas CompositeModes **Canvas supports Porter-Duff style compositing. This scenario applies the composite modes to D2D geometry with a specified clipping and antialiasing behavior.**	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
Direct2D Map **renders a complex set of vectors that comprises a map. The map primarily consists**	300 milliseconds	No glitching at 60 fps	60 FPS

of many path geometries with many 1 pixel wide lines.			
SVG content **there are multiple SVG diagrams that are tested. All the diagrams will be part of one single application. A command line script will cycle through each one of them**	300 milliseconds	No glitching at 60 fps	60 FPS
Text **Tests will include:** **1. Rendering text on full screen.** **2. It cycles between text rendered with anti-alias mode cleartype, grayscale and aliased.** **3. It cycles between text rendering mode 6X1, 6X4, 4X1 and 4X4.** **4. It cycles between 1 glyph run per line, 2 glyph run per line and 4 glyph run per line.**	300 milliseconds	No glitching at 60 fps	60 FPS
Pan and Zoom - Rendering a 21 MegaPixelimage and performing Pan & Zoom operations at the native resolution of the display	300 milliseconds	No glitching at 60fps	60 FPS
Panning through small images **Test will render a set of unique 200 x 200 pixel images at the native resolution of the display, and pan through them in a continuous loop.**	300 milliseconds	No glitching at 60fps	60 FPS

Internet Explorer

• Load and render content from a local on disk cache representing approximately 30 top web sites. • Content representative of dynamic HTML5 + CSS content.	300 milliseconds for a new tab to display with the page content	No glitching at 60 fps	60 FPS

Bandwidth test

The GPU bandwidth is evaluated by measuring the time it takes DWM to compose the client area of a single application with simple content (i.e. no overlap, no glass, etc.) at the native resolution of the panel. The test isn't trying 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 perform 10 (ten) memory I/O operations per pixel @ 60fps

(or 600 * 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	2.4GB/sec
1920x1080	4.8GB/sec

To compute the actual bandwidth, the test will perform following calculation:

Measured Bandwidth (GB/s) = (Number of Pixels in App Client Area) * 8 / DWM_AverageGPUTimePerFrame / (1024*1024)

The test multiplies the number of pixels by 8 (eight) because for each pixel the DWM will read 4 bytes and write 4 bytes (so 8 bytes worth of I/O per pixel).

If the measured bandwidth is greater than the required bandwidth, the test will pass.

Memory Benchmark Test

This is arrived by compositing the results from the following tests:

1. Simple D3D app that breaks at different points (Create_Device, First_Clear, First_Present) to determine the memory cost to reach that point.
2. Resource tests: Several D3D apps each of which creates and uses a different resource
    - Load_Model: Loads a mesh using D3DX and draws it.
    - Upload: copy a system memory texture to the local memory texture several times per frame.
    - Download: Copies the framebuffer to system memory each frame.
    - GDI_Emulation: creates many textures of different sizes and maps each one.

Goal:

- Maximum 1 MB at Create_Device, First_Clear and First_Present
- 4 MB peak for Upload and Download

Design Notes:

If the system has any of the following configurations:

- Multiple GPUs
- 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.

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

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

System.Fundamentals.Graphics.DisplayRender.StableAndFunctional

Target Feature: System.Fundamentals.Graphics.DisplayRender

Title: Display device functions properly and does not generate hangs or faults under prolonged stress

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server x64
- Windows 8 Client ARM
- Windows 7 Client x64
- Windows 7 Client x86
- Windows Server 2008 Release 2 x64

Description:

The system must run under prolonged stress without generating hangs or faults.

Exceptions: Not Specified

Business Justification:

Driver hangs and crashes lead to a negative end user experience.

Scenarios:

Driver reliability.

Success Metric: Pass/Fail

Enforcement Date: Windows 7 RC

Comments:

SYSFUND-8085

System.Fundamentals.Graphics.InternalDisplay

Description:

Base for Graphics on Systems

Related Requirements:

- System.Fundamentals.Graphics.InternalDisplay.NativeResolution

System.Fundamentals.Graphics.InternalDisplay.NativeResolution

Target Feature: System.Fundamentals.Graphics.InternalDisplay

Title: Systems with integrated displays must use native resolution by default

Applicable OS Versions:

- Windows 8 Client ARM
- Windows 8 Client x64
- Windows 8 Client x86
- Windows 7 Client x86
- Windows 7 Client x64

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.

Exceptions: Not Specified

Business Justification:

This requirement ensure that a Windows system can boot into the preferred native resolution without screen flashing or other artifacts

Scenarios:

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.

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

New; Sysfund-8503

System.Fundamentals.Graphics.MultipleDevice

Description:

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

Target Feature: System.Fundamentals.Graphics.MultipleDevice

Title: On a system with multiple graphics adapters, system firmware will allow the user to configure the usage of the adapters

Applicable OS Versions:

- Windows 8 Server x64
- Windows 8 Client x64
- Windows 8 Client x86

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:

1. Enable/Disable any adapter
    a. The firmware must offer the user the ability to select which adapter is enabled or disabled
    b. At any given time at least one adapter, that supports POST, must be enabled

c. If the user enables an adapter, and the system only supports one active adapter at a time, then all other adapters must be disabled

d. 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

2. Select the adapter to be used as POST device

a. Firmware must only allow the user to select one adapter as the POST device.

b. System is allowed to POST only on an adapter that cannot be physically removed from the system.

c. At any given time at least one adapter, that supports POST, must be enabled

d. 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

Exceptions: Not Specified

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

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

New; Sysfund-8080

System.Fundamentals.Graphics.MultipleDevice.SubsystemDeviceID

Target Feature: System.Fundamentals.Graphics.MultipleDevice

Title: Hybrid/Switchable Graphics systems that support multiple discrete graphics adapters or chipset combination must use the same Subsystem ID

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This is required to identify system as being a hybrid GPU through Online Crash Analysis.

Scenarios:

Identify hybrid graphics system through OCA.

Success Metric: Pass/Fail

Enforcement Date: Windows 7

Comments:

SYSFUND-0215

System.Fundamentals.Graphics.RenderOnly

Description:

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

Target Feature: System.Fundamentals.Graphics.RenderOnly

Title: Render Only device on client or server system must be Direct3D 10 capable or greater.

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server 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.

Exceptions: Not Specified

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.

Scenarios:

Following are some example scenarios, this is not an exhaustive list

1. Server Render Farm
    1. User configures a Server system that only contains one GPU installed as a Render only device and does not have any display devices connected to the system.
    2. The user accesses this Server by utilizing Remote Desktop Sessions or alternate means.
    3. Now the user launches a DirectX application that needs to use the advanced computing power of the GPU for some processing (like graphics render farm)
        i. Note: GDI based applications will not work in this case unless, a Full WDDM driver or a Display Only driver is installed along with this OR the user connects via a Remote Desktop session.
    4. The application uses the DirectX APIs to enumerate WDDM GPUs and finds the Render only device and utilizes it for its computational needs
2. Additional GPU for compute
    1. User configures a system with 2 GPUs.
        i. One a full WDDM driver which supports Render and Display functions.
        ii. The second a WDDM render only driver that only support render functions
    2. The user has a monitor connected to the full WDDM graphics hardware and uses it to log into the system

3. Now the user launches an application that needs to use the advanced computing power of the GPU for some processing (like graphics render farm)
4. The application uses the DirectX APIs to enumerate WDDM GPUs and finds both the GPUs.
5. The application selects the Render only device and utilizes it for its computational needs but uses the Full WDDM driver for its UI rendering needs.

Success Metric:

Enforcement Date: Windows 8 RC

Comments:

New;

System.Fundamentals.HAL

Description:

This feature defines Hardware Abstraction Layer (HAL) requirements for systems.

Related Requirements:

- System.Fundamentals.HAL.HPETRequired
- System.Fundamentals.HAL.IfCSRTPresent

System.Fundamentals.HAL.HPETRequired

Target Feature: System.Fundamentals.HAL

Title: System provides a high-precision event timer

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server 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.

Exceptions:

None

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.

Scenarios:

This is to enable multimedia application scenarios.

Success Metric: Not Specified

Enforcement Date: June 1, 2008

Comments: Not Specified

System.Fundamentals.Input

Description:

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

Target Feature: System.Fundamentals.Input

Title: I2C connected HID devices must have a Unique HWID along with a HID compatible ID

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Server x64
- Windows 8 Client ARM

Description:

Description: All I2C connectedHID devices must have a unique HWID and a HID compatible ID that will allow WU to identify the device (when needed) and allowdrivers to be loadedfrom WU.

Design Notes: See Microsoft published HID I2C protocol specification (link not provided yet)

Exceptions:

This requirement is only enforced for HID I2C devices and not generalized for SPB.

Business Justification:

To enable the correct drivers to be loaded by WU

Scenarios:

Input devices connected over I2C

Success Metric: Third party drivers for I2C devices are available on WU

Enforcement Date: Widnows 8 RC

Comments:

New

System.Fundamentals.Input.PS2UniqueHWID

Target Feature: System.Fundamentals.Input

Title: All PS/2 connected devices (such as internal keyboards) must have a unique hardware ID

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server x64

Description:

Description:All PS/2 connecteddevices (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

Exceptions: Not Specified

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

Scenarios:

Internally attached keyboards show the correct and unique HWID and correct compatible ID.

Success Metric: The unique hardware ID should be in a format that allows WU to identify the device and load the correct driver for it. Success is third party touchpad drivers shipping with WU

Enforcement Date: June 1, 2012

Comments: Not Specified

System.Fundamentals.MarkerFile

Description:

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

Target Feature: System.Fundamentals.MarkerFile

Title: System includes marker file

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client ARM
- Windows 8 Server 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.

Exceptions:

None

Business Justification:

This standardizes the format of the marker file.

Scenarios:

Report OCA back to the PC manufacturer.

Success Metric: Pass/Fail

Enforcement Date: Windows 7 RC

Comments:

SYSFUND-0201

System.Fundamentals.Network

Description:

These are system level requirements that may impact the integration with a type of network device.Â

Related Requirements:

- System.Fundamentals.Network.MobileBroadBand
- System.Fundamentals.Network.NetworkListOffloads
- System.Fundamentals.Network.WiFiDirect

System.Fundamentals.Network.MobileBroadBand

Target Feature: System.Fundamentals.Network

Title: Systems that include Broadband support meet Windows requirements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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

Non-USB devices for GSM and CDMA and all WiMax devices (both USB and non-USB based) require IHV developed Mobile broadband drivers. The drivers need to be compliant with the Mobile Broadband Driver Model. IHV developed MB drivers must pass the tests for validating MB functionality, such as the Windows Hardware Certification Kit tests.

- Devices based on the SPI peripheral bus require SPI driver support from the platform vendor.
- USB devices for GSM and CDMA must be compliant with the Microsoft USB driver stack and do not require IHV specific bus driver stack.

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. IHVs can still certify their devices with non-USB bus interfaces. However, Microsoft will not provide any engineering support for non-USB bus interfaces. Device manufacturers must ensure that any non-USB based mobile broadband solution meets all of the mobile broadband certification requirements. Microsoft will be unable to provide any waivers or contingencies for any non-USB bus interfaces towards the certification of non-USB MB solutions.

The following table summarizes the required mobile broadband features.

Attribute	Requirement
Bus	USB-HSIC (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

Exceptions: Not Specified

Business Justification:

Mobile Broadband requirements are needed to ensure high quality Windows scenarios. These also ensures compatibility, interoperability and reliability of devices/drivers when working with windows.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.Network.NetworkListOffloads

Target Feature: System.Fundamentals.Network

Title: Wireless LAN networking device on systems that support Connected Standby must support NDIS 6.30 and support offloads

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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 Stand by 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 TBD)	Yes
Interrupt Moderation	Yes
OS-programmable packet filtering	Yes

Exceptions: Not Specified

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.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.Network.WiFiDirect

Target Feature: System.Fundamentals.Network

Title: Software only uses Wi-Fi direct

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Applications and drivers must only use Wi-Fi direct calls. Drivers and applications must not make use of port 3.

Exceptions: Not Specified

Business Justification:

This is being required to ensure a consistent experience.

Scenarios:

Wi-Fi Direct

Success Metric: Pass/Fail

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.NX

Description:

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

Target Feature: System.Fundamentals.NX

Title: Systems must ship with processors that support NX and include drivers that function normally

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server 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. 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."

Exceptions: Not Specified

Business Justification:

Microsoft Windows offers a number of defensive enhancements designed to protect customers. This technology prevents code from executing in data segments.

Scenarios:

Security

Success Metric: Pass/Fail

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.PowerManagement

Description:

Power management is a feature that turns the PC off or into a lower power state. Requirements in this section describe the requirements around power management.

Related Requirements:

- System.Fundamentals.PowerManagement.MultiPhaseResume
- System.Fundamentals.PowerManagement.PCResumesInTwoSeconds
- System.Fundamentals.PowerManagement.PCSupportsS3S4S5
- System.Fundamentals.PowerManagement.PowerProfile

System.Fundamentals.PowerManagement.MultiPhaseResume

Target Feature: System.Fundamentals.PowerManagement

Title: Storage subsystem supports multi-phase resume from Hibernate

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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

Exceptions:

The boot storage device is booted from a USB bus.

Business Justification:

Significant performance improvement for boot and hibernate resume.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments: Not Specified

System.Fundamentals.PowerManagement.PCResumesInTwoSeconds

Target Feature: System.Fundamentals.PowerManagement

Title: System resumes from ACPI S3 state in less than specified time

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client 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 with dual symmetrical CPU sockets.

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.

Exceptions: Not Specified

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 Tablet PCs, battery life, power management, media center, and small office/home office (SOHO) servers. Fast system startup is also part of the Instant On initiative.

Scenarios:

The ability to have the system Instant On.

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0004

System.Fundamentals.PowerManagement.PCSupportsS3S4S5

Target Feature: System.Fundamentals.PowerManagement

Title: Systems support S3, S4 and S5 states

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server x64
- Windows 8 Client ARM

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.

Exceptions:

None

Business Justification:

The logo requirement is being updated to add support for S4 being required to support new resume scenarios in Windows 8.

Scenarios:

Hiberboot

Success Metric: Not Specified

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0214

System.Fundamentals.PowerManagement.PowerProfile

Target Feature: System.Fundamentals.PowerManagement

Title: System must report form factor via power management profile

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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

Value	Form factor
1	Desktop
2	Mobile/Laptop
6	Appliance PC or All In One
8	Tablet and tablet convertible

Design Notes:

For more information see page 119 of the ACPI specification version 5.0a

Exceptions: Not Specified

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.

Scenarios:

Enables the OS to apply special power profiles based on the form factor.

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.PowerManagement.CS

Description:

Power management is a feature that turns the PC off or into a lower power state. Requirements in this section describes requirement around power management for systems that support connected standby.

Related Requirements:

- System.Fundamentals.PowerManagement.CS.ConnectedStandby
- System.Fundamentals.PowerManagement.CS.ConnectedStandbyDuration
- System.Fundamentals.PowerManagement.CS.CSQuality
- System.Fundamentals.PowerManagement.CS.ResumeFromConnectedStandby

System.Fundamentals.PowerManagement.CS.ConnectedStandby

Target Feature: System.Fundamentals.PowerManagement.CS

Title: System supports the connected standby power profile must set the FACP flags

Applicable OS Versions:

- Windows 8 Client ARM
- Windows 8 Client x64
- Windows 8 Client x86

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.*

Exceptions: Not Specified

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.

Scenarios:

A system that is always connected to the internet.

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.PowerManagement.CS.ConnectedStandbyDuration

Target Feature: System.Fundamentals.PowerManagement.CS

Title: Systems that support Connected Standby must meet Connected Standby duration requirement on battery power description

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client ARM
- Windows 8 Client x64

Description:

Systems that support Connected Standby MUST drain less than 5% of system battery capacity over a 16 hour idle period in the default shipping configuration.

If a system that supports Connected Standby does not ship with a battery, this requirement does not apply.

This requirement is validated by programmatically entering the Connected Standby state and measuring the battery capacity drained (in mWh or mAh) over a 16 hour period. The battery capacity drained (in mWh or mAh) must be less or equal to 5% of the battery design capacity.

The test will reside in the Connected Standby state for a duration less than 16 hours to reduce test execution time. The resulting battery capacity drained will be scaled proportionally to 16 hours before evaluating against the 5% goal.

Exceptions: Not Specified

Business Justification:

User expectation of minimal battery charge across idle time.

Scenarios:

Supporting the connected standby feature.

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.PowerManagement.CS.CSQuality

Target Feature: System.Fundamentals.PowerManagement.CS

Title: Systems that support Connected Standby must meet reliability standards for Runtime Power Management

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

System reliability is critical to the customer experience with a Windows 8 platform.

Scenarios:

Systems that support the Connected Standby feature.

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.PowerManagement.CS.ResumeFromConnectedStandby

Target Feature: System.Fundamentals.PowerManagement.CS

Title: System that supports Connected Standby MUST exit the Connected Standby state within 300ms of the user pressing the power button.

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client ARM
- Windows 8 Client x64

Description:

Connected standby exit latency will be measured from the press of the power button to the touch digitizer is on. The duration of a connected standby resume shall be no longer than 300 milliseconds.

Exceptions: Not Specified

Business Justification:

Most users prefer to leave PCs turned off when not in use to reduce noise and electricity use. However, for new uses at home and in business, the PC must be instantly available to answer the phone, display e-mail, or browse the Web, but be silent and use minimal energy when not in use.

Scenarios:

Resuming from standby.

Success Metric: Pass/Fail

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.PXE

Description:

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

Target Feature: System.Fundamentals.PXE

Title: Remote boot support for PXE complies with BIOS Boot Specification 1.01 or EFI boot manager

Applicable OS Versions:

- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 7 Client x86
- Windows Server 2008 Release 2 x64
- Windows 8 Server 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.

Exceptions:

None

Business Justification:

Provides IT administrators a way of remotely booting and installing images on systems.

Scenarios:

Booting and installing images

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0072

System.Fundamentals.Reliability

Description:

These are for reliability tests content from the former DEVFUND category.

Related Requirements:

- System.Fundamentals.Reliability.SystemReliability

System.Fundamentals.Reliability.SystemReliability

Target Feature: System.Fundamentals.Reliability

Title: Drivers in a system must be reliable

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

This requirement will help ensure a good user experience.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.Security

Description:

Additional filter driver requirements related to security.

Related Requirements:

- System.Fundamentals.Security.TDIAndLSP

System.Fundamentals.Security.TDIAndLSP

Target Feature: System.Fundamentals.Security

Title: No TDI or LSP Filters are installed by the driver or associated software packages during installation or usage

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server x64

Description:

There can be no use of TDI or LSP Filters by either kernel mode software or drivers, or user mode software or drivers.

Design Notes:

Transport Driver Interface (TDI) is a protocol understood by the upper edge of the Transport layer of the Microsoft Windows kernel network stack commonly used to communicate with the various network transport protocols. Alternatives, such as NDIS in some cases, can be used to achieve similar message passing between layers in the network architecture.

Layered Service Provider (LSP) is a feature of the Microsoft Windows Winsock 2 Service Provider Interface (SPI) which it uses to insert itself in the form of a DLL into the TCP/IP stack.

Exceptions: Not Specified

Business Justification:

Use of TDI and LSP filters increase attack surface, and will therefore no longer be supported for future OS releases.

Scenarios: Not Specified

Success Metric: Pass/Fail

Enforcement Date: Windows RC

Comments:

SYSFUND-8531

System.Fundamentals.SignedDrivers

Description:

This feature checks for signed drivers.

Related Requirements:

- System.Fundamentals.SignedDrivers.BootDriverEmbeddedSignature
- System.Fundamentals.SignedDrivers.DigitalSignature

System.Fundamentals.SignedDrivers.BootDriverEmbeddedSignature

Target Feature: System.Fundamentals.SignedDrivers

Title: Boot drivers must be self-signed with an embedded signature

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

Boot drivers must be embedded-signed in order to work properly with the boot process.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: 6/1/2007

Comments:

DEVFUND-0029

System.Fundamentals.SignedDrivers.DigitalSignature

Target Feature: System.Fundamentals.SignedDrivers

Title: System must contain logo qualified devices

Applicable OS Versions:

- Windows 7 Client x86

- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server x64
- Windows 8 Client ARM

Description:

All buses, devices and other components in a system must meet their respective Windows Hardware Certification Requirements 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 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.

In previous updates, this requirement went by SYSFUND-0001 for Windows Vista and SYSFUND-0192 for the Windows 7 logo program.

Exceptions:

none.

Business Justification:

The Windows Logo on the system is an identification of quality and that all components in the system have been tested.

Scenarios:

This requirement satisfies that all devices in the system have passed Windows Logo Program testing.

Success Metric: Pass or fail.

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0001, SYSFUND-0192

System.Fundamentals.SMBIOS

Description:

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

Target Feature: System.Fundamentals.SMBIOS

Title: System firmware support for SMBIOS complies with the SMBIOS specification

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Server x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

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.

Scenarios:

TBA

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0074

System.Fundamentals.StorageAndBoot

Description:

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

Target Feature: System.Fundamentals.StorageAndBoot

Title: Boot Devices in systems that support Connected Standby must meet these requirements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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 4.41+	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.

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

When using eMMC as the primary boot device, the eMMC memory must be hardware partitioned such that the boot critical portion of the UEFI 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:

Feature	Specification
Power	
Max Idle Power	<= 5 mW
Random Performance	
4KB Write IOPs (measured overa 1GB area)	>= 200
4KB Write IOPs (measured over a 10GB area)	>= 50
64KB Write IOPs (measured overa 1GB area)	>= 25
4KB Read IOPs (measured overa 10GB area)	>= 2000
4KB 2:1 read/write mix IOPs (measured overa 1GB area)	>= 500
4KB 2:1 read/write mix IOPs (measured overa 10GB area)	>= 140
Sequential Performance	
Write speed (64KB I/Os) (measured overa 10GB area)	>= 40 MB/s
Write speed (1MB I/Os) (measured overa 10GB area)	>= 40 MB/s
Read speed (64KB I/Os) (measured overa 10GB area)	>= 60 MB/s
Device I/O Latency	
Max Latency	< 500 milliseconds

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.

Exceptions: Not Specified

Business Justification:

Microsoft Windows and application performance

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.StorageAndBoot.EncryptedDrive

Target Feature: System.Fundamentals.StorageAndBoot

Title: If a system ships with an Encrypted Drive, then it must meet the following requirements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client ARM
- Windows 8 Client x64

Description:

SATA/USB/eMMC based boot storage devices **must** support the following security features:

- Trusted commands
- TCG OPAL v2.0 and the following feature set - Single User Mode and Additional Data Store Tables
- IEEE 1667 TCG Storage Silo

Exceptions: Not Specified

Business Justification:

This requirement is a key component of enterprise-readiness of Windows 8 Connected Standby-enabled consumer devices

Scenarios:

Without encryption, consumers will not be able to access corporate mail on Windows 8 mobile devices when the Exchange Server requires device encryption.

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.StorageAndBoot.SATABootStorage

Target Feature: System.Fundamentals.StorageAndBoot

Title: System with SATA boot storage must meet requirements

Applicable OS Versions:

- Windows 8 Client x86

- Windows 8 Client x64
- Windows 8 Client ARM

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

Externally connected SATA devices (eSATA) are not supported for boot storage.

When SATA is used as the primary boot device, to insure 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.

Exceptions: Not Specified

Business Justification: Not Specified

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.SystemAudio

Description:

This section contains all of the audio requirements for PCs

Related Requirements:

- System.Fundamentals.SystemAudio.Audio
- System.Fundamentals.SystemAudio.HardwareVolumeControl
- System.Fundamentals.SystemAudio.HDAudioCodecsFollowPNPGuideLines
- System.Fundamentals.SystemAudio.NoiseOnTheSignal
- System.Fundamentals.SystemAudio.SystemEmploysAntiPop
- System.Fundamentals.SystemAudio.SystemMicArray
- System.Fundamentals.SystemAudio.SystemUsesHDAudioPinConfigs
- System.Fundamentals.SystemAudio.TwoCaptureAndRenderScenarios

System.Fundamentals.SystemAudio.Audio

Target Feature: System.Fundamentals.SystemAudio

Title: Systems contain audio devices that conform to Windows Logo requirements

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64
- Windows Server 2008 Release 2 x64

Description:

Systems need to conform to all audio device requirements

Exceptions:

If audio is implemented in the PC, then this requirement must be met.

Business Justification:

This ensures compatibility with Windows.

Scenarios:

Outstanding sound - I can play my music

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0118

System.Fundamentals.SystemAudio.HardwareVolumeControl

Target Feature: System.Fundamentals.SystemAudio

Title: System with user exposed hardware audio volume control(s) uses approved control method to report status

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86

- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Non-tablet/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 Tablet and Convertible PCs a slider or knob is not acceptable. The volume control button must either be a rocker or two button combinations 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.

Exceptions:

Audio volume buttons are required for Tablet 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.

Scenarios:

Outstanding sound - I can listen to my music.

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0057

System.Fundamentals.SystemAudio.HDAudioCodecsFollowPNPGuideLines

Target Feature: System.Fundamentals.SystemAudio

Title: System with HD Audio codecs follows applicable Plug and Play guidelines

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Systems need to conform to **Device.Audio.HDAudio.PnPCodecDeviceID.**

In addition, the system firmware manages the related registers. The requirements ensure that codec implementations can be uniquely identified.

Exceptions:

If audio is implemented in this system, then this requirement must be met.

Business Justification:

To ensure HD Audio solutions will work on Microsoft Windows PCs

Scenarios:

Outstanding Sound - I can listen to my music.

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0055

System.Fundamentals.SystemAudio.NoiseOnTheSignal

Target Feature: System.Fundamentals.SystemAudio

Title: Noise on the signal from the audio device generated by system components is -80dB FS or better

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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.

Exceptions:

None

Business Justification:

Improving PC audio fidelity is important to support the media usage scenarios.

Scenarios:

Plug into my headphone jack and listen to music

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0048

System.Fundamentals.SystemAudio.SystemEmploysAntiPop

Target Feature: System.Fundamentals.SystemAudio

Title: System employs anti-pop measures on all system audio outputs

Applicable OS Versions:

- Windows 7 Client x86

- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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 implemented in various ways such as by using a system board switch or relay design.

Exceptions:

None

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.

Scenarios:

Outstanding sound, I can plug into my headphone jack and listen to my music.

Success Metric: Pass/Fail

Enforcement Date: Not Specified

Comments:

SYSFUND-0050

System.Fundamentals.SystemAudio.SystemMicArray

Target Feature: System.Fundamentals.SystemAudio

Title: If the system includes microphone array hardware then it exposes the array to the Microsoft class drivers through defined methods

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

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

Exceptions:

If a Microphone array is implemented then it must meet this requirement.

Business Justification:

Reporting the correct geometry to the Windows Operating System will ensure that software processes the audio properly.

Scenarios:

When I plug in a communication device my PC knows what to do with it.

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0067

System.Fundamentals.SystemAudio.SystemUsesHDAudioPinConfigs

Target Feature: System.Fundamentals.SystemAudio

Title: System uses the HD Audio device pin configuration registers to expose logical devices supported by the Windows UAA HD Audio class driver

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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. 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. 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 codecs 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.

Exceptions:

None

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.

Scenarios:

Outstanding sound when I plug into my headphone jack and listen to my music.

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0049

System.Fundamentals.SystemAudio.TwoCaptureAndRenderScenarios

Target Feature: System.Fundamentals.SystemAudio

Title: System includes hardware for at least two capture and render scenarios

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

If a system audio solution implements two or more input jacks (including internal MIC) and two or more output jacks (including internal speaker) there must be at least two stereo DACs and at least two stereo ADCs to support independent behavior for two of the input jack/output jack pairs. At least one logical input port must be independent and at least one logical output port must be independent.

Design Notes:

To enable ubiquitous support for real-time communication from the PC audio hardware, regardless of other media playback or recording occurring at the same time, it is recommended that all primary audio solutions in a system with audio capabilities include at least two analog-to-digital converters and separate connectivity for both. The system would also support multi-streaming defined as multiple audio streams to multiple independent devices as exposed by the UAA-compliant audio device. See the Universal Audio Architecture Hardware Design Guidelines at http://go.microsoft.com/fwlink/?LinkId=50734.

Exceptions:

None

Business Justification:

This requirements aims to enable communication through a certified Windows audio solution at all times, creating a telephone like experience where the PC can be trusted to always be there for communication purposes regardless of what else is going on in the system

Scenarios:

Being able to plug into my headphone jack and listen to music.

Success Metric: Pass/fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0047

System.Fundamentals.SystemAudio.3rdPartyDriver

Description:

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

Target Feature: System.Fundamentals.SystemAudio.3rdPartyDriver

Title: Audio device is compliant with one of the appropriate technology specifications supported by the UAA initiative

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

Audio devices in systems must comply with the Universal Audio Architecture (UAA).

Exceptions: Not Specified

Business Justification:

This helps ensure compatibility with Microsoft Windows Universal Audio Architecture.

Scenarios:

Play great sound.

Success Metric: Not Specified

Enforcement Date: September 17, 2008

Comments: Not Specified

System.Fundamentals.SystemPCIController

Description:

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

Target Feature: System.Fundamentals.SystemPCIController

Title: System devices and firmware meet PCI requirements

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server 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.

Exceptions: Not Specified

Business Justification:

Ensure compatibility with industry standard.

Scenarios:

Compatibility

Success Metric: Pass/Fail

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0109

System.Fundamentals.SystemPCIController.SystemImplementingRiserCard

Target Feature: System.Fundamentals.SystemPCIController

Title: System board implementing a riser card provides a unique ID for the riser

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server 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.

Exceptions: Not Specified

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.

Scenarios:

Device compatibility

Success Metric: Not Specified

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0025

System.Fundamentals.SystemUSB

Description:

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.USBControllerTransferSpeed
- 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.XHCIToEHCIControllerTransitions

System.Fundamentals.SystemUSB.EHCIToXHCIControllerTransitions

Target Feature: System.Fundamentals.SystemUSB

Title: System firmware handles hand off of transition from xHCI USB controller to eHCI USB controller

Applicable OS Versions:

- Windows 8 Client x64
- Windows 8 Client x86
- Windows 8 Server 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.

Exceptions: Not Specified

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.

Scenarios:

Compatibility

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments: Not Specified

System.Fundamentals.SystemUSB.ExternalUSBonCSisEHCIorXHCI

Target Feature: System.Fundamentals.SystemUSB

Title: External USB ports on system that support connected standby must be EHCI or XHCI

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server 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

Exceptions: Not Specified

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.

Scenarios:

This requirement supports the Connected standby feature.

Success Metric: USB Keyboard and mouse works correctly on all externally exposed USB ports and the USB OHCI/UHCI drivers are not loaded.

Enforcement Date: 06/01/2012

Comments:

System.Fundamentals.SystemUSB.SuperSpeedCapableConnectorRequirements

Target Feature: System.Fundamentals.SystemUSB

Title: Each exposed SuperSpeed capable connector supports SuperSpeed, high, full and low speed USB devices routed through its xHCI controller

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Server x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

This requirement ensures that low, full, and high speed devices continue to work as xHCI controllers become more prevalent in systems.

Scenarios:

Device Compatibility

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8019

System.Fundamentals.SystemUSB.SuperSpeedPortsAreVisualDifferent

Target Feature: System.Fundamentals.SystemUSB

Title: Systems with SuperSpeed Ports and non-SuperSpeed ports must have visual differentiation between the two types of ports

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server x64
- Windows 8 Client ARM

Description:

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.

Exceptions: Not Specified

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.

Scenarios:

Device compatibility

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

System.Fundamentals.SystemUSB.SuperSpeedTerminationRemainsOn

Target Feature: System.Fundamentals.SystemUSB

Title: SuperSpeed termination remains on once power is applied to the bus, unless the OS explicitly removes it.

Applicable OS Versions:

- Windows 8 Client x64
- Windows 8 Client x86
- Windows 8 Server x64
- Windows 8 Client ARM

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 connectors 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.

Exceptions: Not Specified

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.

Scenarios:

Compatibility

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments: Not Specified

System.Fundamentals.SystemUSB.SystemExposesUSBPort

Target Feature: System.Fundamentals.SystemUSB

Title: Systems must have at least one user-accessible USB port

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

All systems must expose at least one external USB host port. For the best experience with Windows, all systems 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.

However, if the system does not expose a Standard USB A Port, it must ship with an adapter that changes the port exposed by the system to a Standard USB A Port. 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 Support
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.

Exceptions: Not Specified

Business Justification:

Allowing users to use their existing USB devices is a differentiating feature of Windows. The micro-USB port or proprietary docking port allows thinner form factors to be designed.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.SystemUSB.TestedUsingMicrosoftUsbStack

Target Feature: System.Fundamentals.SystemUSB

Title: Systems with xHCI Controllers must be tested with Microsoft's xHCI Stack installed

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64

Description:

Systems with Extensible Host Controller Interface (xHCI) Controllers must be tested with Microsoft's xHCI Stack installed and enabled.

Exceptions: Not Specified

Business Justification:

All USB host controllers must be compatible with the Microsoft inbox driver stack.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.SystemUSB.USB3andUSB2PortsRoutedToSameXHCIController

Target Feature: System.Fundamentals.SystemUSB

Title: 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

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64

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

Exceptions: Not Specified

Business Justification:

This reduces the risk of a system crash due to fast switch of devices between USB ports.

Scenarios:

Device compatibility and stability.

Success Metric: Pass/Fail

Enforcement Date: December 1, 2010

Comments:

SYSFUND-0225

System.Fundamentals.SystemUSB.USBControllerTransferSpeed

Target Feature: System.Fundamentals.SystemUSB

Title: USB Host Controllers can achieve transfer speeds as defined in the table below

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

xHCI USB Host controllers have the ability to achieve at least the transfer speeds as defined in the table below for their respective type

Operating Speed	Theoretical Limit	Transfer to Achieve	Transfer to Achieve MB/s
USB 3.0	5 Gb/s	2.4 Gb/s	300 MB/s

Exceptions: Not Specified

Business Justification:

This targets 3rd Party controller drivers which are used with USB 3.0. Though these speeds are in the specifications they are not tested by the USB-IF.

Scenarios:

Performance

Success Metric: Pass/Fail

Enforcement Date: December 1, 2010

Comments:

SYSFUND-0236

System.Fundamentals.SystemUSB.USBDevicesandHostControllersWorkAfterPowerCycle

Target Feature: System.Fundamentals.SystemUSB

Title: All USB devices and host controllers work properly upon resume from sleep, hibernation or restart without a forced reset of the USB host controller

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

This registry value was created as a fix for devices that were not coming back upon resume.

Scenarios:

Power Management

Success Metric: Pass/Fail

Enforcement Date: December 1, 2010.

Comments:

SYSFUND-0235

System.Fundamentals.SystemUSB.XhciBiosHandoffFollowsSpec

Target Feature: System.Fundamentals.SystemUSB

Title: xHCI BIOS handoff follows specification

Applicable OS Versions:

- Windows 8 Client x64
- Windows 8 Client x86
- Windows 8 Server x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

Driver software expects the BIOS to perform handoff compliant to specification.

Scenarios:

Compatibility

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments: Not Specified

System.Fundamentals.SystemUSB.xHCICompatibleUnlessForApprovedTargetDesigns

Target Feature: System.Fundamentals.SystemUSB

Title: System that support Connected Standby must implement SuperSpeed ports as defined in the xHCI and SuperSpeed specification or use approved eHCI reference design

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client ARM
- Windows 8 Client x64

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.

Exceptions: Not Specified

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.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.SystemUSB.XHCIControllerSaveState

Target Feature: System.Fundamentals.SystemUSB

Title: xHCI controllers correctly save and restore state, or else indicates an error

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 7 Client x64
- Windows 7 Client x86
- Windows 8 Client ARM

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)

Exceptions: Not Specified

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 WHCK, and publish this requirement to highlight the importance of the error mechanism for save/restore state.

Scenarios:

USB Host controllers work seamlessly through power transition states.

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8010

System.Fundamentals.SystemUSB.XHCIControllersMustHaveEmbeddedInfo

Target Feature: System.Fundamentals.SystemUSB

Title: Systems with xHCI controllers must have embedded ACPI information for port routing

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows Server 2008 Release 2 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 )
```

Exceptions: Not Specified

Business Justification:

We need this info to be correct so we know which of the available ports are SuperSpeed capable. Well use this to direct the user to the correct port when they connect a SuperSpeed device.

Scenarios:

Device compatibility

Success Metric: Pass/Fail

Enforcement Date: December 1, 2010

Comments:

SYSFUND-0226

System.Fundamentals.SystemUSB.xHCIControllerSupportMSIInterrupts

Target Feature: System.Fundamentals.SystemUSB

Title: xHCI Controllers support MSI and/or MSI-X interrupts

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client 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.

Exceptions: Not Specified

Business Justification:

This is to ensure compliance with the industry specification

Scenarios:

Device compatibility

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8017

System.Fundamentals.SystemUSB.XhciSupportsMinimum31Streams

Target Feature: System.Fundamentals.SystemUSB

Title: xHCI controller must support at least 31 primary streams per endpoint

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server x64
- Windows 8 Client ARM

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.

Exceptions: Not Specified

Business Justification:

USB based storage devices must be USB Attached SCSI Protocol (UASP) and will utilize streams to achieve faster data transfer rates.

Scenarios: Not Specified

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

System.Fundamentals.SystemUSB.XHCIToEHCIControllerTransitions

Target Feature: System.Fundamentals.SystemUSB

Title: Once the power is applied to the bus, the SuperSpeed termination remains on unless the OS explicitly removes it

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x64
- Windows 8 Client x86

- Windows 8 Server 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 connectors 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.

Exceptions: Not Specified

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.

Scenarios:

Compatibility

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

System.Fundamentals.TrustedPlatformModule

Description:

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.ConnectedStandby
- System.Fundamentals.TrustedPlatformModule.SupportSecureStartUpInPreOS
- System.Fundamentals.TrustedPlatformModule.TPM20
- System.Fundamentals.TrustedPlatformModule.TPMComplieswithTCGTPMMainSpecification
- System.Fundamentals.TrustedPlatformModule.TPMEnablesFullUseThroughSystemFirmware
- System.Fundamentals.TrustedPlatformModule.TPMRequirements
- System.Fundamentals.TrustedPlatformModule.Windows7SystemsTPM

System.Fundamentals.TrustedPlatformModule.ConnectedStandby

Target Feature: System.Fundamentals.TrustedPlatformModule

Title: Connected Standby systems must implement TPM v2.0

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64

Description:

MANDATORY. Connected Standby systems must implement a TPM2.0 solution that meets the requirements called out in **System.Fundamentals.TrustedPlatformModule.TPM20**.

MANDATORY. All platforms that implement a TPM must ensure invariance of PCRs 0, 2, 4 and 7 across power cycles in the absence of changes to the platforms static core 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.

Exceptions: Not Specified

Business Justification:

This requirement is needed to support TPM auto-provisioning, BitLocker and Measured Boot with Attestation functionality; key features of Windows.

Scenarios:

Secure Boot, Measured Boot, BitLocker

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

TPM 2.0 is a new technology; follow up with your Windows Certification contact directly for more information on how to get access to the proprietary specifications called out above, including Microsoft's "TPM v2.0 Command and Signature Profile,".

System.Fundamentals.TrustedPlatformModule.SupportSecureStartUpInPreOS

Target Feature: System.Fundamentals.TrustedPlatformModule

Title: Systems support secure startup by providing system firmware support for writing to and reading from USB flash devices in the pre-operating system environment

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Server x64
- Windows 8 Client ARM

Description:

MANDATORY: 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, formatted with any one of the Microsoft-supported file systems including FAT16, FAT32, EXFAT, or NTFS.

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.

Exceptions: Not Specified

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.

Scenarios:

BitLocker Secure startup and Recovery

Success Metric: Pass/Fail

Enforcement Date: Windows 7 RC

Comments:

SYSFUND-0069

System.Fundamentals.TrustedPlatformModule.TPM20

Target Feature: System.Fundamentals.TrustedPlatformModule

Title: Requirements for all systems that implement a TPM 2.0

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64

Description:

For a system that implements TPM 2.0[1], the platform must comply with the following requirements:

1. **MANDATORY**. 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 Environment[3]	***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[2].	X	
	Secure (isolated from runtime OS) storage of: a. Values (such as an endorsement primaryseed) that survive complete platform power off as well as firmware updates; b. 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 reset to a random value); and c. Values (such as the Platform ConfigurationRegisters) that survive platform power-down to the equivalent of ACPI S3 if TPM2_Shutdown(TPM_SU_STATE) is called but	X	

	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 PCRs0 through 23 for SHA-1, dedicated to the same boot measurements as TPM 1.2.	X	
	• Support for SHA-1 and RSA-2048 algorithms.	X	
	• Robustness against side channel attacks including Differential Power Analysis (DPA) and Electromagnetic Emanations (EM)		X

2. **MANDATORY.** 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.

Measured Boot

Measured Boot allows a 3rd party to validate the compliance of firmware and software loaded during boot with regards to a policy.

1. **MANDATORY**. 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.

2. **MANDATORY**. 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 measured boot log will 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 Measured Boot. The interface to the signature ("attestation") mechanism shall comply with the requirements defined in Microsoft Corporation, Trusted Execution Environment ACPI Profile, Draft .9 dated December 9th, 2011.[4]

3. **MANDATORY**. The system shall include a trusted execution environment supporting the command set defined in Microsoft Corporation, "TPM v2.0 Command and Signature Profile, Draft .9 dated December 9th, 2011."[4]

4. **MANDATORY**. The system shall support the interface specified in Microsoft Corporation, Trusted Execution Environment ACPI Profile, Draft .9 dated December 9th, 2011.[4]

5. **MANDATORY**. The system shall support the interface and protocol specified in Microsoft Corporation, "Trusted Execution Environment EFI Protocol, Draft .9 dated December 9th, 2011,"[4] with the exception of the PCR[7] measurements specified in Appendix A of said document.

6. **OPTIONAL**. The system should support measurements into PCR [7] as specified in Appendix A of the previously referenced Microsoft Corporation, "Trusted Execution Environment EFI Protocol, Draft .9 dated December 9th, 2011" document provided that all mandatory requirements from **System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby** are met and any firmware update that rolls back to an earlier insecure firmware version either clears the TPM state or is not permitted. If these requirements are not met, the referenced Appendix A measurements for PCR [7] **must not** be implemented. This requirement applies to previous TPM versions as well.

7. **MANDATORY**. All platforms that implement a TPM must ensure invariance of PCRs 0, 2 and 4 across power cycles in the absence of changes to the platforms static core 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.

8. **MANDATORY**. An Endorsement Key (EK) unique for each device shall be provisioned during manufacturing. Once provisioned, the EK shall be stored in protected storage accessible only to a trusted execution environment[3].

9. **MANDATORY.** If the provisioned EK cannot be replaced, then the system shall also be provisioned (during manufacturing) with an EK certificate associated with the EK. The EK certificate profile will be provided separately.

10. **OPTIONAL.** If the provisioned EK can be replaced post-manufacturing, the manufacturer is still recommended to provision an EK certificate associated with the EK.

[1]Microsoft uses the term TPM 2.0 for what is also known as TCG TPM.Next.

[2]Note: This requirement may also be met through sealed storage/sealed key blobs that, upon unseal, are held in isolated storage.

[3]A TPM 1.2 or a TPM 2.0 is an example of a Trusted Execution Environment.

[4]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.

Exceptions: Not Specified

Business Justification:

This requirement is needed to support TPM auto-provisioning, BitLocker and Measured Boot with Attestation functionality; key features of Windows.

Scenarios:

Secured Boot, Measured Boot, BitLocker

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments:

TPM 2.0 is a new technology; follow up with your Windows Certification contact directly for more information on how to get access to the proprietary specifications called out above, including Microsoft's "TPM v2.0 Command and Signature Profile," April 2011.

System.Fundamentals.TrustedPlatformModule.TPMCompliesswithTCGTPMMainSpecification

Target Feature: System.Fundamentals.TrustedPlatformModule

Title: 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.

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

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.

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 TPMs 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 a 24 hour time period before entering the first level of defense. Alternately, the default system image must contain non-default software anti-hammering settings which correspond to TPM default behavior. 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.

Note: Windows uses more TPM functionality than previous releases so Windows Certification Tests for the TPM are more extensive.

Exceptions: Not Specified

Business Justification:

This requirement is needed to support BitLocker and Measured Boot with Attestation functionality, key features of Windows.

Scenarios:

BitLocker

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8009

System.Fundamentals.TrustedPlatformModule.TPMEnablesFullUseThroughSystemFirmware

Target Feature: System.Fundamentals.TrustedPlatformModule

Title: Systems with Trusted Platform Modules enable full use of the TPM through system firmware enhancements

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Server 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.

Design Notes:

For information on the role of system firmware in the chain of trust for the SRTM, see the TCG PC Client Specific Implementation Specification for Conventional BIOS, available at http://go.microsoft.com/fwlink/?LinkId=58380 and the TCG PC Client Specific Physical Presence Interface Specification Version 1.00, dated July 13, 2005 or later, and the TCG Platform Reset Attack Mitigation Specification, Revision 0.92 or later, available to TCG Member companies at

http://go.microsoft.com/fwlink/?LinkId=58379. (Use of the TCG Platform Reset Attack Mitigation Specification Version 1.00 or later is recommended and is publicly available at http://go.microsoft.com/fwlink/?LinkId=58380) Also see Windows Vista BitLocker Client Platform Requirements, dated May 16, 2006, or later, available at http://go.microsoft.com/fwlink/?LinkId=70763.

Exceptions:

If a TPM is implemented in the system, then the requirement must be met.

Business Justification:

TPM hardware is only functional when paired with fully TCG compliant system firmware. One of the required features to deliver a more trustworthy computing environment is to leverage the capabilities of a platform with a TPM and to do this very early in the boot process to assure the integrity and entropy of the system components and keys.

Scenarios:

BitLocker

Success Metric: Pass/Fail

Enforcement Date: Windows 7 RC

Comments:

SYSFUND-0031

System.Fundamentals.TrustedPlatformModule.TPMRequirements

Target Feature: System.Fundamentals.TrustedPlatformModule

Title: System implementing TPM must meet requirements

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server x64
- Windows 8 Client ARM

Description:

For a system that implements a Trusted Platform Module (TPM) 1.2, the platform must comply with the following specifications:

1. The TCG Platform Reset Attack Mitigation Specification Version 1.00, Revision 1.00 or later, http://www.trustedcomputinggroup.org/files/temp/6452209B-1D09-3519-AD815636FC36C5CF/Platform%20Reset%20Attack%20Mitigation%20Specification.pdf.

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 Specific TPM Interface Specification (TIS) Version 1.21 Revision 1.00

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 TCG PC Client Specific Implementation Specification for Conventional BIOS, Version 1.20, Revision 1.00 or later. (Note: The errata version 1.21 is strongly recommended instead.)

6. The TCG ACPI Specification Version 1.00, Revision 1.00, http://www.trustedcomputinggroup.org/resources/server_work_group_acpi_general_specification_version_10.

And the system MUST meet the following additional requirements:

7. The Windows Vista BitLocker Client Platform Requirements, dated May 16, 2006, or later, available at http://go.microsoft.com/fwlink/?LinkId=70763.

8. **MANDATORY**. All platforms that implement a TPM must ensure invariance of PCRs 0, 2 and 4 (and 7, if implemented) across power cycles in the absence of changes to the platforms static core 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.

- 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.)
- The platform must implement the memory mapped I/O interface for the TPM (e.g. port based I/O is not sufficient)
- 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, 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.)
- 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.

- 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.
- 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 will help facilitate automated testing of the physical presence interface during Windows Hardware Certification Kit testing and will permit managed environments to completely automate TPM management from the OS without physical presence if an enterprise decides to set the NoPPIClear flag.)
- 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.
- 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.
- 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).
- 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.)
- 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.)
- 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.

- 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.)
- The system firmware must support clearing the TPM from within a setup menu.
- At least 256 bytes of TPM NVRAM must be reserved (and available) for OS use.
- The firmware must issue the TPM_ContinueSelfTest during boot such that the self-test completes before the OS loader is launched.

- 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.)

- 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.

- The platform may or may not issue the TPM_ContinueSelfTest upon resume from S3 (sleep).
- 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.
- The platform should support measurements into PCR [7] as specified in Appendix A of Microsoft Corporation, "Trusted_Execution_Environment_EFI_Protocol, Draft .9 dated December 9th, 2011"[1], provided that all mandatory requirements from System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby are met and any firmware update that rolls back to an earlier insecure firmware version either clears the TPM state or is not permitted. If these requirements are not met, Appendix A measurements for PCR [7] must not be implemented. If the appendix A measurements for PCR[7] are implemented, the platform must successfully pass the BitLockerend-to-end compatibility test of TPM only with PCR[7,11]; reboot and hiberboot.

[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.

Exceptions: Not Specified

Business Justification:

This requirement is needed to support TPM auto-provisioning, BitLocker and Measured Boot with Attestation functionality, key features of Windows.

Scenarios:

BitLocker

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8008

System.Fundamentals.TrustedPlatformModule.Windows7SystemsTPM

Target Feature: System.Fundamentals.TrustedPlatformModule

Title: Systems with Trusted Platform Modules use TPM v1.2, or later

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 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 (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.

Exceptions: Not Specified

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.

Scenarios:

BitLocker

Success Metric: Pass/Fail

Enforcement Date: Windows 7 RC

Comments:

SYSFUND-0030

System.Fundamentals.USBBoot

Description:

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

Target Feature: System.Fundamentals.USBBoot

Title: System firmware supports booting from all exposed USB 2.x, and 3.x ports

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Server x64

Description:

System BIOS or UEFI firmware must by default:

- Support booting through USB 2.x on eHCI controllers
- Support booting through USB 2.x, and 3.x on xHCI controllers.
- Support these on all exposed USB port up to a hub depth of 1 from the exposed-end hub.

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.

All systems deployed in UEFI-mode must support both section 9 and section 3.1 of UEFI 2.3.1. All systems deployed in BIOS-mode must offer boot from UFD as a configuration option.

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

This requirement is unrelated to the default boot order. Bootable USB devices do not have to be enumerated before booting from other devices (e.g. SATA).

Additionally, USB only needs to be enumerated during POST if:

- USB is set to be used for the next boot, or
- No other boot entries higher in the boot order are found and bootable, or
- There is no permanently attached PS2 keyboard.

For example, a compliant system must behave as follows: If the only connected disk is a bootable USB drive (assume appropriate disk configurations), the system should boot from that drive with NO user intervention.

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.

Exceptions: Not Specified

Business Justification:

Booting from USB has been a certificatoin 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.

Scenarios:

Support the ability to boot from a USB thumb drive.

Success Metric: Pass or Fail

Enforcement Date: 1/1/2010

Comments:

SYSFUND-0070

System.Fundamentals.USBBoot.SupportSecureStartUpInPreOS

Target Feature: System.Fundamentals.USBBoot

Title: Systems support secure startup by providing system firmware support for writing to and reading from USB flash devices in the pre-operating system environment

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows 8 Client x86
- Windows 8 Client x64

Description:

A system that will support BitLocker recovery and strong authentication scenarios must support specific system firmware enhancements that enable full-speed reading and writing of data (such as key backup and recovery information) from and to a USB mass storage class device, formatted with any one of the Microsoft-supported file systems including FAT16, FAT32, or NTFS.

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.

Exceptions:

None.

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.

Scenarios:

Secure startup.

Success Metric: Pass/Fail

Enforcement Date: Windows 7 RC

Comments:

SYSFUND-0069

System.Fundamentals.USBBootInternal

Description:

The feature and requirements are about being able to boot from a USB device.

Related Requirements:

- System.Fundamentals.USBBootInternal.USBBootDiskMustBootFromUSB3UASPDrive
- System.Fundamentals.USBBootInternal.USBStorage

System.Fundamentals.USBBootInternal.USBBootDiskMustBootFromUSB3UASPDrive

Target Feature: System.Fundamentals.USBBootInternal

Title: Systems with a USB primary boot disk must boot from a USB 3.0 UASP compatible drive.

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM

Description:

If the primary system boot device is connected to the USB bus, the boot device must be USB 3.0 UASP compliant and connected to an XHCI controller through an internal port. Additionally, the USB boot drive must report itself as a FIXED device rather than removable.

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: http://msdn.microsoft.com/en-us/library/ff553550(v=VS.85).aspx

It is recommended that the boot device be connected directly to an xHCI root hub port rather than through an intermediate hub. On an ARM system it is required that the USB boot device be connected directly to an xHCI root hub port.

Exceptions: Not Specified

Business Justification:

USB 3.0 desktop penetration is expected to be 80% or more by 2012 (IDC 2008) USB 3.0 mobile penetration is expected to be 45% or more by 2012 (IDC 2008)USB 3.0 provides 5 Gbps bandwidth, comparable with SATA3.UASP supports parallel transfer streams with much lower latency for random reads/writes than the USB Mass Storage Protocol. The UASP transport provides up to 30% greater throughput than the BOT transport on USB 3.0 protocol.

Scenarios:

USB Boot

Success Metric: Pass/Fail

Enforcement Date: Windows 8 RC

Comments:

SYSFUND-8016

System.Fundamentals.USBBootInternal.USBStorage

Target Feature: System.Fundamentals.USBBootInternal

Title: USB Storage Devices must use xHCI

Applicable OS Versions:

- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Client ARM
- Windows 8 Server x64

Description:

If a storage device on a system is connected via USB the US bus must support xHCI.

Host Controller Interface (HCI)	System / Boot Device	Data Disk Device
OHCI/UHCI	No	No
EHCI (High Speed Devices)	No	Yes
XHCI (Super Speed Device Only)	Yes	Yes

Exceptions: Not Specified

Business Justification:

Microsoft Windows and application performance.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.USBDevice

Description:

These requirements apply to USB devices that are integrated into a system.

Related Requirements:

- System.Fundamentals.USBDevice.SelectiveSuspend

System.Fundamentals.USBDevice.SelectiveSuspend

Target Feature: System.Fundamentals.USBDevice

Title: All internally connected USB devices must support selective suspend by default

Applicable OS Versions:

- Windows 8 Client x86

- Windows 8 Client x64
- Windows 8 Client ARM

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 users 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.

Exceptions: Not Specified

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."

Scenarios: Not Specified

Success Metric: On certified systems, all internal USB devices enter selective suspend within 60 seconds of being idle.

Enforcement Date: Windows RC

Comments: Not Specified

System.Fundamentals.WatchDogTimer

Description:

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

Target Feature: System.Fundamentals.WatchDogTimer

Title: If a Watch Dog Timer is implemented and exposed through a WDRT, it must meet Windows compatibility and functionality requirements

Applicable OS Versions:

- Windows 7 Client x86
- Windows 7 Client x64
- Windows Server 2008 Release 2 x64
- Windows 8 Client x86
- Windows 8 Client x64
- Windows 8 Server 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

Exceptions:

None

Business Justification:

Ensures compatibility with the Windows Operating System.

Scenarios:

This is to ensure that if a watch dog timer is implemented, it meets compatibility requirements.

Success Metric: Not Specified

Enforcement Date: June 1, 2006

Comments:

SYSFUND-0033

System.Server.Base

Description:

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

Target Feature: System.Server.Base

Title: A server system can natively run a 64-bit version of Windows Server

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0021

System.Server.Base.BMC

Target Feature: System.Server.Base

Title: Baseboard management controller solution must meet requirements

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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].

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2009

Comments:

SYSFUND-0112

System.Server.Base.BMCDiscovery

Target Feature: System.Server.Base

Title: Baseboard Management Controller is discoverable and enumerable

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0078

System.Server.Base.Compliance

Target Feature: System.Server.Base

Title: Server system includes components and drivers that comply with Windows Hardware Certification Program

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0054

System.Server.Base.DevicePCIExpress

Target Feature: System.Server.Base

Title: Server system includes storage and network solutions that use PCI Express architecture

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/fail

Enforcement Date: 6/1/2008

Comments:

SYSFUND-0115

System.Server.Base.ECC

Target Feature: System.Server.Base

Title: System memory uses ECC or other technology to prevent single-bit errors from causing system failure

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0024

System.Server.Base.essentials

Target Feature: System.Server.Base

Title: Windows Storage Server 2008 R2 Essentials system meets requirements.

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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:

- 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).
- 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:

- In normal operation, the primary hard drive must be the first boot device.
- 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

- 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.
- 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.

System must meet Windows Hardware Certification Program driver 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:

- DVD and Blu-Ray optical drives allowed for media playback (HDMI) and other applications
- OEMs can choose to ship slot load or tray load optical drives
- Optical drive is required to use SATA
- 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:

- Wireless NIC must be IEEE 802.11 compliant
- Recommended configuration: 802.11n
- No restrictions on antenna design/implementation
- A server Dashboard based wireless configuration application must be provided
- A wired network adapter is required for installation.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server.

Scenarios:

Windows Server Essentials

Success Metric: Pass/Fail

Enforcement Date: 6/1/2011

Comments:

SYSFUND-0232

System.Server.Base.HotPlugECN

Target Feature: System.Server.Base

Title: Server system that supports native Hot Plug functionality meets requirements defined in Hot-Plug ECN No. 31

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 8/31/2007

Comments:

SYSFUND-0184

System.Server.Base.NoPATA

Target Feature: System.Server.Base

Title: Persistent storage devices on servers classified as Hard Disk Drives must not be PATA

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2009

Comments:

SYSFUND-0209

System.Server.Base.OSInstall

Target Feature: System.Server.Base

Title: Server system includes a method for installing the operating system for emergency recovery or repair

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0010

System.Server.Base.PCI23

Target Feature: System.Server.Base

Title: PCI or PCI-X devices in a server system comply with PCI Local Bus Specification, Revision 2.3 unless otherwise noted

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 x64

Description:

Server system includes only PCI 2.3 compliant PCI or PCI-X devices unless exemptions are noted by an individual requirement.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/fail

Enforcement Date: 6/1/2008

Comments:

SYSFUND-0115

System.Server.Base.PCIAER

Target Feature: System.Server.Base

Title: Windows Server systems may implement AER (Advanced Error Reporting) as specified in PCI Express Base Specification version 2.1 and ACPI Specification 3.0b

Applicable OS Versions:

- Windows 8 Server x64

Description:

This is an If Implemented requirement for Server systems. PCI Express in Server systems may implement the MCFG ACPI table in PCI Firmware Specification, Revision 3.0b, so that the operating system can access the extended configuration space AER Capabilities Register information. If this is implemented, the following requirements must be met:

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

Design Notes:

There is no requirement to provide AER_OSC to give control to the operating system, and systems may implement a firmware first error policy.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Not Specified

Enforcement Date: 1/1/2012

Comments: Not Specified

System.Server.Base.RemoteManagement

Target Feature: System.Server.Base

Title: Server system supports remote, headless, out of band management capabilities

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0035

System.Server.Base.ResourceRebalance

Target Feature: System.Server.Base

Title: Server device drivers must support Resource Rebalance requests

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2007

Comments:

SYSFUND-0142

System.Server.Base.ServerRequiredComponents

Target Feature: System.Server.Base

Title: Server system must include necessary devices and functionality

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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	Policy TBD prior to RTM	Maximum supported memory
	TBD by server performance team prior to RTM	Minimum supported memory
	System.Server.Base.ECC	ECC
Processor	System.Server.Virtualization.ProcessorVirtualizationAssist	Hyper-V support
	TBD by server performance team prior to RTM	Minimum supported processor speed
	Policy TBD prior to 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	Examples; DVD, PXE
Debug	System.Fundamentals.DebugPort.SystemExposesDebugInterface	Examples; COM, USB, 1394, etc.
Remote & OOB Mngmt	System.Server.Base.RemoteManagement	Examples; BMC, Service Processor adapter, etc.
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

a. Bus Controllers & Ports

i. HD Audio

ii. Cardbus & PCMCIA

iii. IEEE 1394

iv. Secure Digital

b. Connectivity

i. Bluetooth

ii. Cardbus & PCMCIA

iii. ExpressCard

iv. IEEE 1394

v. Infrared

vi. Parallel [sysfund-0221] & Serial

vii. Wireless USB

c. Network

i. ISDN

ii. TCP Chimney NIC

d. Display

i. Auxiliary Displays

e. Input

i. Smart Card Reader

f. Streaming Media & Broadcast

i. Broadcast Receiver

ii. Decoder

iii. Encoder

iv. Video Capture

g. TPM

i. TPM. If implemented, must meet requirements.

ii. USB write for BitLocker Recovery

h. Watchdog Timer (WDT)

i. Baseboard Management Controller (BMC)

j. Enhanced Power Management Additional Qualification

k. Dynamic Partitioning Additional Qualification,

l. Fault Tolerance Additional Qualification

m. High Availability Additional Qualification

n. Power Management concerning S3, S4 and S5 system states support

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2007

Comments:

SYSFUND-0129

System.Server.Base.SystemPCIExpress

Target Feature: System.Server.Base

Title: Server system supports PCI Express natively

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/fail

Enforcement Date: 6/1/2008

Comments:

SYSFUND-0115

System.Server.DynamicPartitioning

Description:

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

Target Feature: System.Server.DynamicPartitioning

Title: Servers that support hardware partitioning must supply partition management software as a Windows application running on a Windows operating system.

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Not Specified

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0102

System.Server.DynamicPartitioning.ApplicationInterface

Target Feature: System.Server.DynamicPartitioning

Title: Servers that support hardware partitioning must supply partition management software that provides a GUI and a scripting capability for partition management

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Not Specified

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0105

System.Server.DynamicPartitioning.ConfigurationPersist

Target Feature: System.Server.DynamicPartitioning

Title: Servers that support hardware partitioning must support persistence of hardware partition configuration information across a reboot and power cycle

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0098

System.Server.DynamicPartitioning.Core

Target Feature: System.Server.DynamicPartitioning

Title: Systems that support Dynamic Hardware Partitioning must meet requirements

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 x64

Description:

Systems must meet the requirements listed below and pass the Dynamic Hardware Partitioning test in the Windows Hardware Certification 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

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 12/1/2007

Comments:

SYSFUND-0184

System.Server.DynamicPartitioning.ErrorEffect

Target Feature: System.Server.DynamicPartitioning

Title: Errors detected in a hardware partition on servers that support hardware partitioning cause no operating system-detectable effects on other partitions

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0097

System.Server.DynamicPartitioning.Firmware

Target Feature: System.Server.DynamicPartitioning

Title: Servers that support hardware partitioning must provide server description and partitioning flows in firmware that comply with the Dynamic Hardware Partitioning Requirements Specification

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Not Specified

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0103

System.Server.DynamicPartitioning.HotAddLocal

Target Feature: System.Server.DynamicPartitioning

Title: 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

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0096

System.Server.DynamicPartitioning.HotAddReplace

Target Feature: System.Server.DynamicPartitioning

Title: Servers that support hardware partitioning must support hot addition of processors, memory, and I/O and hot replace of processor and memory subsystems

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Not Specified

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0107

System.Server.DynamicPartitioning.HotAddVisual

Target Feature: System.Server.DynamicPartitioning

Title: Servers that support hardware partitioning must provide visual user indication of the status of hot-add events if no software-based notification is provided

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Not Specified

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0108

System.Server.DynamicPartitioning.HotReplacePU

Target Feature: System.Server.DynamicPartitioning

Title: In servers that support dynamic partitioning, hot replacement PUs must have equal and compatible hardware resources to the PU being replaced

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Not Specified

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0101

System.Server.DynamicPartitioning.PartialHotAdd

Target Feature: System.Server.DynamicPartitioning

Title: Partial success of a hot-add action on a server that supports dynamic partitioning does not affect the stability of the partition or server

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Not Specified

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0100

System.Server.DynamicPartitioning.SoftwareStatus

Target Feature: System.Server.DynamicPartitioning

Title: Servers that support hardware partitioning must supply partition management software that provides the user with status for each hot-add or hot-replace event

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Not Specified

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0106

System.Server.DynamicPartitioning.Subsystem

Target Feature: System.Server.DynamicPartitioning

Title: On servers that support dynamic partitioning, I/O subsystems are provided in a different partition unit to processors and memory subsystems

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Not Specified

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0099

System.Server.FaultTolerant

Description:

This feature defines fault tolerant requirements of server systems

Related Requirements:

- System.Server.FaultTolerant.Core

System.Server.FaultTolerant.Core

Target Feature: System.Server.FaultTolerant

Title: Systems supporting Fault Tolerant operations must meet requirements

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 12/1/2007

Comments:

SYSFUND-0190

System.Server.Firmware.UEFI.GOP

Description:

This section describes requirements for systems implementing UEFI firmware.

Related Requirements:

- System.Server.Firmware.UEFI.GOP.Display

System.Server.Firmware.UEFI.GOP.Display

Target Feature: System.Server.Firmware.UEFI.GOP

Title: System firmware must support GOP and Windows display requirements

Applicable OS Versions:

- Windows 8 Server x64

Description:

If the system firmware supports UEFI, it may choose 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

1. 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.
2. In case multiple displays are detected, UEFI must display the Pre-OS screen based on the following logic
    a. System with an Integrated display(Laptop, All In One, Tablet): UEFI must display the Pre-OS screen only on the integrated display
    b. 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

1. 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
    a. System with an Integrated display(Laptop, All In One, Tablet): The display must always be set to its native resolution and native timing
    b. System **without** an Integrated display (desktop):

i. UEFI must attempt to set the native resolution and timing of the display by obtaining it from the EDID.

ii. If that is not supported, UEFI must select an alternate mode that matches the same aspect ratio as the native resolution of the display.

iii. At the minimum, UEFI must set a mode of 1024 x 768

iv. If the display device does not provide an EDID, UEDI must set a mode of 1024 x 768

c. The firmware must always use a 32 bit linear frame buffer to display the Pre-OS screen

d. PixelsPerScanLine must be equal to the HorizontalResolution.

e. PixelFormat must be PixelBlueGreenRedReserved8BitPerColor. Note that a physical frame buffer is required; PixelBltOnly is not supported.

Mode Pruning

1. 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

1. 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.

    a. 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

    b. If the display is not integrated and does not have an EDID, then the UEFI does not need to manufacture an EDID

Exceptions: Not Specified

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.

Scenarios:

1. Power up a laptop/portable device
    a. In this case, the firmware will determine the native resolution of the integrated display and display the pre-OS screen at the native resolution
2. Power up a desktop with multiple monitors connected
    a. In this case, the firmware will determine the best display to enable and then use the native resolution of the display
3. In case the 3rd party WDDM driver is disabled, the, Microsoft Basic Display Driver will be running. This driver will only allow the user to use the resolution that the system was in before the Microsoft Basic Display driver was run.

Success Metric: Not Specified

Enforcement Date: Windows 8 RC

Comments:

For client systems BIOS mode is only allowed for systems that ship with 32 bit Windows. This allows OEMs to maintain one firmware image for both 64 bit and 32 bit deployments. All systems must be UEFI mode capable and also pass 64 bit UEFI logo requirements.

System.Server.Firmware.VBE

Description:

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

Target Feature: System.Server.Firmware.VBE

Title: System firmware that supports VBE must comply with the Windows Display requirements

Applicable OS Versions:

- Windows 8 Server 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

1. 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.

2. In case multiple displays are detected, video device firmware must display the Pre-OS screen based on the following logic:

    a. 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

    b. 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

1. 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

    a. System with an Integrated display(Laptop, All In One, Tablet/Convertible): The display must always be set to its native resolution and native timing

    b. System **without** an Integrated display (desktop):

        i. The video device firmware must attempt to set the native resolution and timing of the display by obtaining it from the EDID

        ii. 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.

        iii. At the minimum, the video device firmware must set a mode of 1024 x 768

iv. If the display device does not provide an EDID, UEDI must set a mode of 1024 x 768

c. The video device firmware must always use a 32 bit linear frame buffer to display the Pre-OS screen

d. PixelsPerScanLine must be equal to the HorizontalResolution.

e. PixelFormat must be PixelBlueGreenRedReserved8BitPerColor. Note that a physical frame buffer is required; PixelBltOnly is not supported.

Mode Pruning

1. 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

2. 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

3. The video device firmware must support 800 x 600 and 1024 x 768

4. All modes must be progress at 60 Hz

Providing the EDID

1. 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

    a. 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

    b. If the display is not integrated and does not have an EDID, then the video device firmware does not need to manufacture an EDID

Exceptions: Not Specified

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.

Scenarios:

Power up a laptop/portable device.

In this case, the firmware will determine the native resolution of the integrated display and display the pre-OS screen at the native resolution.

Power up a desktop with multiple monitors connected.

In this case, the firmware will determine the best display to enable and then use the native resolution of the display Microsoft Basic Display driver is installed on POST device on a system with a BIOS.

In case the 3rd part WDDM driver is disabled, the, Microsoft Basic Display Driver will be running. This driver will only allow the user to set those resolutions that are enumerated by the video device firmware based on the mode pruning against the EDID of the display

Success Metric:

Enforcement Date: Windows 8 RC

Comments:

System.Server.Graphics

Description:

Base for Graphics on Server Systems

Related Requirements:

- System.Server.Graphics.WDDM

System.Server.Graphics.WDDM

Target Feature: System.Server.Graphics

Title: All Windows graphics drivers must be WDDM

Applicable OS Versions:

- Windows 8 Server 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.

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	D3D9, D3D10, D3D10.1, D3D11	GDI Hardware acceleration, Connecting and configuring

	2008 R2		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:

a. 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.

b. 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.

c. 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

Exceptions: Not Specified

Business Justification:

See description

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

New; Sysfund-8096

System.Server.Graphics.XDDM

Description:

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

Target Feature: System.Server.Graphics.XDDM

Title: Server system graphics solution is based on XDDM unless 3D acceleration is supported

Applicable OS Versions:

- 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.

Exceptions: Not Specified

Business Justification:

If 3D acceleration is supported the graphics driver must be based on WDDM to increase stability of the system.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

System.Server.HighAvailability

Description:

This feature defines High Availability requirements of server systems

Related Requirements:

- System.Server.HighAvailability.Core

System.Server.HighAvailability.Core

Target Feature: System.Server.HighAvailability

Title: Systems earning the High Availability Additional Qualification must meet requirements

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 x64

Description:

Minimum Hardware Features and Vendor Requirements for Windows High Availability Additional Qualification:

System or Chassis+Blade(s) is Windows Server Logo Qualified

System, or Blade in Chassis, runs a 64-bit version of the Windows ServerAv operating system or Windows Server Virtualization

System, or Blade in Chassis, is 2 Socket-capable

System, Blade or Chassis must include N+1 Hot Swap Power Supplies, or have a power supply for each blade, such that the failure of a single blade power supply would not affect any other blade or component of the chassis or frame

System Blade or Chassis must include Hot Swap Fans, or have a fan(s) for each blade, such that the failure of a single fan would not affect any other blade or component of the chassis or frame

System, or Blade in Chassis, does not include any ATA, ATAPI or IDE ports or hard drives

System, or Blade in Chassis, includes RAID solution for Storage

Any RAID technology is acceptable and Customer preference can determine whether software RAID, a RAID adapter, or a Storage Array which includes RAID, is used

System or Chassis includes Failure Alert Indicators for fans, power supplies, system or processor temperature overages, and system voltage overages

Any solution to alert the user of a problem and isolate it is acceptable, such as a Baseboard Management Controller [BMC], Service Processor, or physical indicators such as lights

System, or Blade in Chassis, includes Multi-Path IO Storage [MPIO] solution, or a Redundant IO path

System, or Blade in Chassis, includes Load Balancing and Fail Over [LBFO] Network solution [also known as teaming], or a Redundant IO path

System or Blades in Chassis include memory failure protection that exceeds standard ECC ability to correct a single bit error and detect a double bit error.

Examples of memory protection technologies that meet this requirement include Mirror or RAID 1 RAM, RAID 5 RAM, Single or Double Device Data Correction [SDDC or DDDC], On-Line or Hot Spare, or ECC methods that would allow correction of double bit or greater errors.

Vendor must uniquely identify the HA system by name or model designation, and not allow any of the above components to be unselected from the component list of the system

Vendor must be a licensee of the operating system

Vendor must be a Gold level Partner

Vendor must have a Partner Advantage Support agreement with Microsoft

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 12/1/2007

Comments:

SYSFUND-0191

System.Server.PowerManageable

Description:

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

Target Feature: System.Server.PowerManageable

Title: Power manageable servers support the power metering and budgeting ACPI interface

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2009

Comments:

SYSFUND-0207

System.Server.PowerManageable.PerformanceStates

Target Feature: System.Server.PowerManageable

Title: If processor(s) in a server system support performance states, the server provides mechanisms to makes these states available to Windows

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2006

Comments:

SYSFUND-0008

System.Server.PowerManageable.RemotePowerControl

Target Feature: System.Server.PowerManageable

Title: Power manageable server provides a standards based remote out-of-band interface to query and control the power of the system

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 x64

Description:

Power manageable server provides an out of band remote management interface from the servers 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.

More detail on the SMASH profile can be found on the Distributed Management task Force web site at http://go.microsoft.com/fwlink/?LinkId=237147

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: Not Specified

Comments:

SYSFUND-0208

System.Server.RemoteFX

Description:

This feature defines RemoteFX requirements of server systems

Related Requirements:

- System.Server.RemoteFX.RemoteFX

System.Server.RemoteFX.RemoteFX

Target Feature: System.Server.RemoteFX

Title: Server systems supporting RemoteFX must meet requirements

Applicable OS Versions:

- Windows 8 Server 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 1.2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports the RemoteFX OS feature.

Scenarios: Not Specified

Success Metric: Not Specified

Enforcement Date: Not Specified

Comments:

Win7 Sysfund-0228

System.Server.SMBIOS

Description:

This feature defines SMBIOS requirements of server systems

Related Requirements:

- System.Server.SMBIOS.SMBIOS

System.Server.SMBIOS.SMBIOS

Target Feature: System.Server.SMBIOS

Title: System firmware must fully and accurately implement SMBIOS structures of type 16 and of type 17

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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): 1. Physical Memory Array (Type 16) containing information on Location, Use and Error Correction Types; pages 51-52. 2. 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. 3.

Memory Device (Type 17) containing information on Form Factor, Type and Type Detail; pages 52-54 4. 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. 5. 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: 1. 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. 2. Allow server and data center management solutions to exploit this information in their diagnostic tools and methods for better RAS abilities. 3. Enable certain classes of ISV products (RAM disk, etc.) to exploit this information for better performance and functionalities on Windows platforms.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2009

Comments:

SYSFUND-0210

System.Server.SVVP

Description:

This feature defines requirements for the SVVP program

Related Requirements:

- System.Server.SVVP.SVVP

System.Server.SVVP.SVVP

Target Feature: System.Server.SVVP

Title: Products participating in the Server Virtualization Validation Program must meet requirements

Applicable OS Versions:

- Windows Server 2008 Release 2 x64
- Windows 8 Server x64

Description:

Server platforms participating in the Server Virtualization Validation Program must meet the requirements called out here: http://www.windowsservercatalog.com/svvp.aspx.

Exceptions: Not Specified

Business Justification:

Virtualization is a key element of Windows Server.

Scenarios:

Server virtualization

Success Metric: Pass/Fail

Enforcement Date: 1/1/2011

Comments:

SVVP is not a certification program. This requirement is called out to facilitate inclusion in the WHCK.

System.Server.SystemStress

Description:

This feature defines system stress requirements of server systems

Related Requirements:

- System.Server.SystemStress.ServerStress

System.Server.SystemStress.ServerStress

Target Feature: System.Server.SystemStress

Title: Server system must function correctly under stress

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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

Exceptions: Not Specified

Business Justification:

This requirement is critical for server RAS, which is critical for the Windows Server platform.

Scenarios:

Server RAS

Success Metric: This will be tested using stress tools that emulate loads which may be placed upon a Windows Server system.

Enforcement Date: 1/1/2011

Comments: Not Specified

System.Server.Virtualization

Description:

This feature defines virtualization requirements of server systems

Related Requirements:

- System.Server.Virtualization.ProcessorVirtualizationAssist

System.Server.Virtualization.ProcessorVirtualizationAssist

Target Feature: System.Server.Virtualization

Title: Processors in the server support virtualization hardware assists

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2007

Comments:

SYSFUND-0140

System.Server.WHEA

Description:

This feature defines WHEA requirements of server systems

Related Requirements:

- System.Server.WHEA.Core

System.Server.WHEA.Core

Target Feature: System.Server.WHEA

Title: Server enables reporting of system hardware errors to the operating system

Applicable OS Versions:

- Windows 8 Server x64
- Windows Server 2008 Release 2 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.

Exceptions: Not Specified

Business Justification:

This requirement supports Windows Server RAS, which is a key tenet of Windows Server platforms.

Scenarios:

Windows Server RAS

Success Metric: Pass/Fail

Enforcement Date: 6/1/2009

Comments:

SYSFUND-0014