Keyboard Enhancements in Windows 8

September 28, 2012

Abstract

Over the last decade, there has been a trend in the industry toward adding consumer control buttons (CCB) to the standard keyboard. These additional consumer control buttons generally represent multimedia, web browsing, application launch, and power management capabilities, and are reported via the Human Interface Device (HID) device class. A limited subset of these buttons will continue to be supported over legacy PS/2 keyboards (Set 1) primarily for legacy laptops. Laptops based on System-on-Chip (SoC) design must migrate to HID (leveraging transports like USB or I2C).

This paper is primarily intended for input device independent hardware developers (IHVs) and original equipment manufacturers (OEMs) and suggests solutions for implementing keyboards (external and internal) on Windows 8. It outlines the opportunities, challenges and benefits along with references to additional information. It assumes that the reader is familiar with details on the HID stack for Windows as a building block for additional enhancements in Windows 8.

This information applies to the following operating systems:   
 Windows 8

Windows Server 2012

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

The current version of this paper is maintained on the Web at:   
 [Keyboard Enhancements to Windows 8](http://msdn.microsoft.com/en-us/library/windows/hardware/hh975382)

**Disclaimer**: This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet website references, may change without notice. Some information relates to pre-released product which may be substantially modified before it’s commercially released. Microsoft makes no warranties, express or implied, with respect to the information provided here. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

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

© 2012 Microsoft. All rights reserved.

C:\Users\jenlin\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Outlook\KN5ONHWU\dep_MicrosoftLogotype.png

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| September 28, 2012 | Updated to reflect rebranding | | | |
| May 6, 2012 | First publication | | | |

Contents

[Introduction 3](#_Toc335223743)

[Support for Consumer Control Buttons 3](#_Toc335223744)

[Launcher-specific AppCommands 5](#_Toc335223745)

[Browser-specific AppCommands 6](#_Toc335223746)

[Media-specific AppCommands 6](#_Toc335223747)

[Power-specific HID Usages 7](#_Toc335223748)

[Wireless Radio Control Button/Switch 7](#_Toc335223749)

[Screen Brightness Control 8](#_Toc335223750)

[HID over I2C 8](#_Toc335223751)

[Introduction to the Architecture 9](#_Toc335223752)

[Best Practices for Integrated Keyboards on Mobile PCs 9](#_Toc335223753)

[Best Practices for Keyboards on Docking Solutions 10](#_Toc335223754)

[Windows Hardware Certification Requirements for Input 10](#_Toc335223755)

[Call to Action for Enhanced Keyboard Features 11](#_Toc335223756)

[Resources 11](#_Toc335223757)

[Appendix 1: Report Descriptors for Enhanced HID Keyboard 13](#_Toc335223758)

[Keyboard TLC 13](#_Toc335223759)

[Consumer Control and Power TLC 14](#_Toc335223760)

[Appendix 2: AppCommands Remapping 16](#_Toc335223761)

[Appendix 3: Implementing Airplane Mode Radio Management Switch/Button/LED 17](#_Toc335223762)

[Software Solution (for Existing Systems) 18](#_Toc335223763)

[Guidance for PC Manufacturers 18](#_Toc335223764)

[Best Practices for Third-Party Drivers 18](#_Toc335223765)

[Installation Scenarios 20](#_Toc335223766)

[Hardware Solution (for New Systems) 21](#_Toc335223767)

[Sample Report Descriptors 21](#_Toc335223768)

[Validating Airplane Mode 23](#_Toc335223769)

# Introduction

Over the last decade, there has been a trend in the industry toward adding enhanced features to the standard keyboard. These new features--typically consumer control buttons (CCBs) related to multimedia, web browsing, application launch, and power management capabilities--represent a departure from the traditional use of a simple keyboard.

In addition, keyboards on mobile form factor PCs have traditionally been attached over a PS/2 connector. As mobile platforms migrate to System-on-Chip (SoC) form factor designs, additional innovation is provided for keyboards over a new transport. The Human Interface Device (HID) device class is extended in Windows 8 to work over I2C. This paper describes the opportunities and benefits as keyboards on laptop and docking solutions migrate away from PS/2.

Finally this paper provides references to the Windows Hardware Certification Kit (HCK), and includes guidance and motivation around this program. It ends with a summary of the calls to action and references for further reading.

**NOTE:** The functionalities of the CCBs listed in this paper for Windows 8 are identical to x86/64 and Windows on Arm, unless specifically called out in a particular section.

# Support for Consumer Control Buttons

The Windows 2000 operating system introduced support for a limited set of CCBs for keyboards. A limited set of these buttons were also supported on PS/2 (Set 1). These CCBs continue to be supported in Windows 8. The user experience has been enhanced for a number of them and is summarized below. Along with these CCBs, some additional buttons have been added for airplane mode and screen brightness support.

For a complete list of current HID usages that are appropriate for enhanced multimedia, web browsing, and application launch functionality, see the Consumer Page (Page 0x0C) in the latest [HID Usage Tables specification](http://www.usb.org/developers/hidpage/) and associated approved review requests. The list of supported HID and PS/2 (Scan Set 1) CCBs is contained in the following tables. Please note that the support for the regular keyboard buttons (from the Keyboard usage page) continue to be supported as well.

Table 1. HID consumer page usages supported in Windows 8

|  |  |  |
| --- | --- | --- |
| ****Usage**** | ****Usage name**** | ****Usage type**** |
| 0xB5 | Scan Next Track | One Shot Control (OSC) |
| 0xB6 | Scan Previous Track | One Shot Control (OSC) |
| 0xB7 | Stop | One Shot Control (OSC) |
| 0xCD | Play/Pause | One Shot Control (OSC) |
| 0x6F | Brightness Increment | Re-trigger Control (RTC) |
| 0x70 | Brightness Decrement | Re-trigger Control (RTC) |
| 0xE0 | Volume | Linear Control (LC) |
| 0xE2 | Mute | On/Off Control (OOC) |
| 0xE3 | Bass | Linear Control (LC) |
| 0xE4 | Treble | Linear Control (LC) |
| 0xE5 | Bass Boost | On/Off Control (OOC) |
| 0xE9 | Volume Increment | Re-trigger Control (RTC) |
| 0xEA | Volume Decrement | Re-trigger Control (RTC) |
| 0x0152 | Bass Increment | Re-trigger Control (RTC) |
| 0x0153 | Bass Decrement | Re-trigger Control (RTC) |
| 0x0154 | Treble Increment | Re-trigger Control (RTC) |
| 0x0155 | Treble Decrement | Re-trigger Control (RTC) |
| 0x018A | AL Email Reader | Selector (Sel) |
| 0x0221 | AC Search | Selector (Sel) |
| 0x0223 | AC Home | Selector (Sel) |
| 0x0224 | AC Back | Selector (Sel) |
| 0x0225 | AC Forward | Selector (Sel) |
| 0x0226 | AC Stop | Selector (Sel) |
| 0x0227 | AC Refresh | Selector (Sel) |
| 0x022A | AC Bookmarks | Selector (Sel) |
| 0x0192 | AL Calculator | Selector (Sel) |
| 0x0194 | AL Local Browser | Selector (Sel) |
| 0x0183 | AL Consumer Control Configuration | Selector (Sel) |

Table 2. PS/2 scan codes supported in Windows

|  |  |  |
| --- | --- | --- |
| ****Key name**** | ****Set 1**** ****Make**** | ****Set 1**** ****Break**** |
| Scan Next Track | E0 19 | E0 99 |
| Scan Previous Track | E0 10 | E0 90 |
| Stop | E0 24 | E0 A4 |
| Play/Pause | E0 22 | E0 A2 |
| Mute | E0 20 | E0 A0 |
| Volume Increment | E0 30 | E0 B0 |
| Volume Decrement | E0 2E | E0 AE |
| AL Email Reader | E0 6C | E0 EC |
| AC Search | E0 65 | E0 E5 |
| AC Home | E0 32 | E0 B2 |
| AC Back | E0 6A | E0 EA |
| AC Forward | E0 69 | E0 E9 |
| AC Stop | E0 68 | E0 E8 |
| AC Refresh | E0 67 | E0 E7 |
| AC Bookmarks | E0 66 | E0 E6 |
| AL Calculator | E0 21 | E0 A1 |
| AL Local Browser | E0 6B | E0 EB |
| AL Consumer Control Configuration | E0 6D | E0 ED |

In Windows, HIDServ (a SVCHost Service) converts these HID usages to AppCommands Window Messages and injects them to the application with focus. The PS/2 driver also sends the corresponding Virtual Keys (VKs) for the PS/2-associated CCBs. If the application in focus fails to process the AppCommand VK, it is sent to the shell (where default functionality is registered).

AppCommands are broken in to the following categories:

* Launcher-specific AppCommands
* Browser-specific AppCommands
* Media-specific AppCommands
* Application-generic AppCommands

## Launcher-specific AppCommands

The following table compares the experiences associated with the launcher-specific AppCommands for Windows 7 and Windows 8.

Table 3. Launcher-specific AppCommands

|  |  |  |  |
| --- | --- | --- | --- |
| Consumer control button | WM\_AppCommand | Windows 7 | Windows 8 |
| Mail | APPCOMMAND\_LAUNCH\_MAIL = 15 | Launches a new session of user’s registered Mail client | Launches a new session of user’s registered default Mail client (either on Desktop or registered Windows Store app). |
| My Computer | APPCOMMAND\_LAUNCH\_APP1 = 17 | Launches a new session of “My Computer” or registered app | Launches a new session of “My Computer” or registered app **on Desktop** |
| My Calculator | APPCOMMAND\_LAUNCH\_APP2 = 18 | Launches a new session of “Calc” or registered app | Launches a new session of “Calc” or registered app **on Desktop** |

**Note**: Launch browser and media player are covered in the specific sections below.

Windows 8 supports the paradigm of remapping these CCBs. For more information about remapping, please see the Appendix.

## Browser-specific AppCommands

The following table compares the experiences associated with the browser-specific AppCommands in Windows 7 and Windows 8. Internet Explorer 10 has almost identical CCB support as Internet Explorer 10 for the desktop.

Table 4. Browser-specific AppCommands

|  |  |  |  |
| --- | --- | --- | --- |
| Consumer control button | WM\_AppCommand | Windows 7\* | Windows 8\* |
| Browser Back | APPCOMMAND\_BROWSER\_BACKWARD = 1 | Explorer Back | Works on both Internet Explorer 10 and Internet Explorer 10 for the desktop. |
| Browser Forward | APPCOMMAND\_BROWSER\_FORWARD = 2 | Explorer Forward | Works on both Internet Explorer 10 and Internet Explorer 10 for the desktop. |
| Browser Refresh | APPCOMMAND\_BROWSER\_REFRESH = 3 | Explorer Refresh | Works on both Internet Explorer 10 and Internet Explorer 10 for the desktop. |
| Browser Stop | APPCOMMAND\_BROWSER\_STOP = 4 | N/A | Works on both Internet Explorer 10 and Internet Explorer 10 for the desktop. |
| Search | APPCOMMAND\_BROWSER\_SEARCH = 5 | Explorer Search (App search vs. Windows search) | Launches the Search pane |
| Favorites | APPCOMMAND\_BROWSER\_FAVORITES = 6 | Shows Favorites pane | Works only on Internet Explorer 10 for the desktop.\* |
| Browser Home | APPCOMMAND\_BROWSER\_HOME = 7 | Launches browser and takes you to Home page | Works on both Internet Explorer 10 and Internet Explorer 10 for the desktop. |

*\*These CCBs are being investigated for support in Internet Explorer 10.*

## Media-specific AppCommands

The following table compares the experiences associated with the media-specific AppCommands in Windows 7 and Windows 8. The functionality is identical.

Table 5. Media-specific AppCommands

|  |  |  |  |
| --- | --- | --- | --- |
| Consumer control button | WM\_AppCommand | Windows 7 | Windows 8 |
| Next Track | APPCOMMAND\_MEDIA\_NEXTTRACK = 11 | Works as expected | Works as expected |
| Previous Track | APPCOMMAND\_MEDIA\_PREVIOUSTRACK = 12 | Works as expected | Works as expected |
| Play/Pause | APPCOMMAND\_MEDIA\_PLAY\_PAUSE = 14 | Works as expected | Works as expected |
| Stop | APPCOMMAND\_MEDIA\_STOP = 13 | Works as expected | Works as expected |
| Volume Mute | APPCOMMAND\_VOLUME\_MUTE = 8 | Works as expected | Works as expected |
| Volume Down | APPCOMMAND\_VOLUME\_DOWN = 9 | Works as expected | Works as expected |
| Volume Up | APPCOMMAND\_VOLUME\_UP = 10 | Works as expected | Works as expected |
| Media | APPCOMMAND\_LAUNCH\_MEDIA\_SELECT = 16 | Works as expected | Launches a new session of user’s registered default media client (either on Desktop or registered Windows Store app) |

**Note**: Windows store apps use the Windows.Media.MediaControls class to get the functionality defined by most CCBs. This class raises events to the application when the buttons are pressed (with the exception of Volume and Mute, which are handled directly by the operating system).

## Power-specific HID Usages

Windows 8 continues to provide support for the Sleep CCB. There is no Windows AppCommand associated with the Sleep button. It must be noted that the behavior of the Sleep button varies based on the platform configuration in Windows 8.

Table 6. Power-specific HID usages

|  |  |  |  |
| --- | --- | --- | --- |
| Consumer control button | WM\_AppCommand | Windows 7\* | Windows 8\* |
| Sleep | N/A | Works as expected (Standby-S3) | * SOC = Connected Standby * Non-SoC = Standby (S3 default but user configurable) |

*\*Users can map via power management Control Panel (Desktop) to ignore when it is pressed.*

## Wireless Radio Control Button/Switch

Windows 8 has added new support and standardized a solution for keyboards (external or embedded on laptops) to identify a Wireless Radio control (for example, Airplane mode) button/switch through HID. Windows 8 supports the HID usages listed in the following table.

Table 7. HID usages on generic desktop page supported in Windows 8

|  |  |  |
| --- | --- | --- |
| ****Usage ID**** | ****Usage name**** | ****Usage type**** |
| 0x000C | Wireless Radio Controls | Collection Application (CA) |
| 0x00C6 | Wireless Radio Button | On/Off Control (OOC) |
| 0x00C7 | Wireless Radio LED | On/Off Control (OOC) |
| 0x00C8 | Wireless Radio Slider Switch | On/Off Control (OOC) |

For more information, please see [Appendix 3](#_Appendix_3_–).

**Note:** These HID usages operate only on mobile systems (battery powered) and require Windows 8 (build 8302 or greater).

## Screen Brightness Control

Windows 8 has added new support and standardized a solution for keyboards (external or embedded on laptops) to control a laptop’s or slate’s screen brightness through HID. The HID committee recently approved [HID Review Request 41](http://www.usb.org/developers/hidpage#Approved_Usage_Table_Review_Requests).

PC manufacturers building laptops or clamshell form factor systems should use the HID Usages defined in [Table 1](#_Support_for_Consumer) as part of their consumer controls collection. Windows 8 supports the HID usages listed in the following table.

Table 8. HID usages on consumer controls page supported in Windows 8

|  |  |  |
| --- | --- | --- |
| ****Usage ID**** | ****Usage name**** | ****Usage type**** |
| 0x006F | Brightness Increment | Re-trigger Control (RTC) |
| 0x0070 | Brightness Decrement | Re-trigger Control (RTC) |

**Note:** These HID usages operate only on mobile systems (battery powered) and require Windows 8 (build 8321 or greater).

# HID over I2C

Windows 8 introduces a new set of transports collectively called Simple Peripheral Buses (SPB). These buses are initially available on SoC form factor systems but may extend to other systems over time and primarily target I2C. To reduce the need for additional software for each system, Windows 8 also enhances the list of supported HID transports by adding support for HID over I2C natively in-box. The HID protocol is the choice (with in-box class drivers) for devices like keyboards, mice, touchpads, speaker phones, buttons, and touchscreens, and to new categories like sensors potentially as they evolve.

Microsoft has been working on an industry standard for HID over I2C. This standard allows devices like sensors, keyboards, and touchpads to work natively over HID as long as the firmware in the device is compliant with this specification.

## Introduction to the Architecture

The following simplified software stack diagram provides an overview of the new HID transport for I2C.

Figure 1: HID I2C Stack Diagram

Key points of interest:

1. SoC silicon partners provide the I2C controller driver (as shown in the fourth and lowest layer, shaded red, in the stack diagram).
2. OEM/ODM accurately represents ACPI I2C capabilities in BIOS (not represented in the stack diagram).
3. Windows 8 provides in-box support for the necessary driver and completes the end-to-end scenario (as shown in the upper three layers, shaded blue, in the stack diagram).

## Best Practices for Integrated Keyboards on Mobile PCs

This section summarizes the key principles for keyboards as they relate to integrated SoC solutions.

* PC manufacturers (OEMs/ODMs) who design embedded keyboards must ensure that the ACPI references are correctly set up to enumerate the keyboard correctly to the operating system.
* PC manufacturers (OEMs/ODMs) who design embedded keyboards on a convertible system must ensure that they correctly represent the presence and availability of the keyboard. Doing so incorrectly will lead to the software keyboard being incorrectly launched or suppressed.
* PC manufacturers (OEMs/ODMs) who design embedded keyboards must ensure that the system power manages the keyboard to conserve power while in connected standby mode but still ensure that the system resumes on key press.
* PC manufacturers (OEMs/ODMs) who design embedded keyboards must ensure that the OEM Keyboard INF correctly identifies the keyboard Type/SubType for localized systems. To ensure correct INF identification, a unique hardware ID is required for Japanese and English localized keyboards, to correctly match in the INF.
* PC manufacturers (OEMs/ODMs) who design embedded keyboards with CCBs beyond those supported by Windows 8 must ensure that they:
* Employ one or more vendor-defined consumer control top-level collections in their HID Descriptor (see the example in Appendix 1).
* Employ a unique hardware ID to allow for the necessary software (that enables the vendor-specific CCBs) to be downloaded from Windows Update (WU).

Additional information is available in the Windows Engineering Guide (WEG) for both x86 and ARM.

## Best Practices for Keyboards on Docking Solutions

This section summarizes the key principles for keyboards as they relate to SoC solutions that employ a docking solution leveraging I2C inside the system. The PC manufacturer is requested to follow the best practices from the sections above (that apply once docked) along with the best practices summarized here:

* PC manufacturers (OEMs/ODMs) who design dockable keyboards must ensure that they correctly identify keyboard or mouse presence to software (for example, identifying when the dock is attached or detached).
* PC manufacturers (OEMs/ODMs) who design dockable keyboards must ensure that they correctly identify keyboard and mouse availability to software (for example, when the keyboard or mouse is accessible to the user).
* PC manufacturers (OEMs/ODMs) who design dockable keyboards must eliminate protocol errors introduced by the docking connector (removed in firmware, where possible).
* PC manufacturers (OEMs/ODMs) who design multiple localized dockable keyboards must ensure that the docking station supports a unique ACPI hardware ID for the keyboard type and subtype associated with each docking solution.

For more information, see the Windows Engineering Guide (WEG) for both x86 and ARM.

# Windows Hardware Certification Requirements for Input

The Windows Hardware Certification Program and Kit (HCK) are integral for ensuring a strong user experience around input devices. For Windows 8, we’ve made targeted enhancements to the Input section of the Windows Hardware Certification Program and Kit to address customer and partner feedback. The main goals were to add value to the certification program, enabling customers to have a simple, quick, and intuitive device experience and to reduce support costs for partners.

The following list shows the key areas of investment in the enhanced input requirements:

1. Laptop-compatible IDs for embedded devices must be unique.
2. Customers can use a device immediately after plug-in, without needing to download additional software.
3. Third-party filter drivers are of high quality and easily available on Windows Update, where possible.
4. Support costs are reduced for hardware partners for external and internal keyboard and mice solutions.
5. Successful docking solutions and experiences are ensured.
6. Upgrade experience for customers upgrading to Windows 8 is improved.
7. All keyboards and mice work with class driver, and the Input requirements enable partners to have higher quality third-party drivers.

**Note:** KVM certification requirements have been deprecated from the Windows Hardware Certification Program for Input Devices due to the extremely low submissions numbers. For information about manual KVM tests, see [Display Guidelines for KVM Switches in Windows 7](http://download.microsoft.com/download/7%2FE%2F7%2F7E7662CF-CBEA-470B-A97E-CE7CE0D98DC2/KVM-Win7.docx).

For additional information on the enhancements made to the input Windows Hardware Certification Program and Kits, please refer to the paper identified in the Resources section.

# Call to Action for Enhanced Keyboard Features

* Keyboard IHVs should implement the design guidelines for HID keyboards described in this article.
* PC manufacturers (OEMs/ODMs) who design embedded keyboards should implement the design guidelines for HID keyboards described in this article.
* PC manufacturers (OEMs/ODMs) building SoC-based systems should design their keyboards and touchpads to leverage HID. Do not leverage PS/2.
* PC manufacturers (OEMs/ODMs) building systems that leverage HID over I2C should ensure that their BIOS can talk to HID devices over I2C, when appropriate.
* PC manufacturers (OEMs/ODMs) with existing systems that have a radio management slider or button should support the software solution identified in [Appendix 3](#_Solution_#1_–).
* All partners should review the Windows Hardware Certification Program and Kit for HID devices and ensure that their solution is in accord the HCK tests.

# Resources

USB Device Class Specifications for Human Interface Devices (HID)

<http://www.usb.org/developers/hidpage/>

HID Audio Controls and Windows

<http://msdn.microsoft.com/en-us/library/windows/hardware/gg462991>

Keyboard Scan Code Specification

<http://msdn.microsoft.com/en-us/library/windows/hardware/gg463372>

Input Device Class Power Management Reference Specification

<http://www.microsoft.com/whdc/resources/respec/specs/pmref/PMinput.mspx>

Input Windows Hardware Certification Program and Kit Enhancements

Targeted for MSDN in 2012

Windows Engineering Guide (x86/64 and ARM)

Draft available on Connect.

# Appendix 1: Report Descriptors for Enhanced HID Keyboard

The following design sample is for a HID keyboard with CCBs defined by Windows and additional vendor-specific CCBs. The keyboard contains the following HID top-level collections (TLC), making it a complex HID Device:

1. Keyboard TLC (Usage Page: Generic Desktop / Usage: Keyboard)
2. Consumer Control TLC (Usage Page: Consumer / Usage: Consumer Controls)
3. Power TLC (Usage Page: Generic Desktop / Usage: System Controls)

**Note:** On some transports, like USB, the first collection is often kept on a separate USB interface to maintain compatibility with some legacy PC BIOS. This is not needed for newer systems or for newer transports such as I2C.

## Keyboard TLC

The following report descriptor represents the keyboard TLC. It is analogous to the HID report descriptor for the boot protocol keyboards to help maximize compatibility with a PC BIOS.

Usage Page (Generic Desktop)

Usage (Keyboard)

Collection (Application)

Report ID (1)

Report Size (1)

Report Count (8)

Usage Page (Keyboard)

Usage Minimum (Keyboard LeftControl)

Usage Maximum (Keyboard RightGUI)

Logical Minimum (0)

Logical Maximum (1)

Input (Data, Variable, Absolute)

Report Count (1)

Report Size (8)

Input (Constant)

;

;Output - LEDs

Report Count (5)

Report Size (1)

Usage Page (LEDs)

Usage Minimum (Num Lock)

Usage Maximum (Kana)

Output (Data, Variable, Absolute)

Report Count (1)

Report Size (3)

Output (Constant)

;

;Input – QWERTY Keys Array

Report Count (6)

Report Size (8)

Logical Minimum (0)

Logical Maximum (255)

Usage Page (Keyboard)

Usage Minimum (0)

Usage Maximum (255)

Input (Data, Array)

End Collection

This TLC issues an 8-byte input report and a 1-byte report ID. The format (with the exception of the Report ID) is identical to the standard keyboard boot protocol report as documented in the HID Class Version 1.11 specification. The interface also allows the host system to turn the respective LED state indicators on and off as appropriate, as specified by the 1-byte output report.

Table 9. Keyboard input report

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ****Byte**** | ****Bit 7**** | ****Bit 6**** | ****Bit 5**** | ****Bit 4**** | ****Bit 3**** | ****Bit 2**** | ****Bit 1**** | ****Bit 0**** |
| **0** | Report ID = 1 | | | | | | | |
| **1** | Right GUI | Right Alt | Right Shift | Right Control | Left GUI | Left Alt | Left Shift | Left Control |
| **2** | Reserved (Constant) | | | | | | | |
| **3** | Keyboard Usage #1 | | | | | | | |
| **4** | Keyboard Usage #2 | | | | | | | |
| **5** | Keyboard Usage #3 | | | | | | | |
| **6** | Keyboard Usage #4 | | | | | | | |
| **7** | Keyboard Usage #5 | | | | | | | |
| **8** | Keyboard Usage #6 | | | | | | | |

Table 10. Keyboard output report

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ****Byte**** | ****Bit 7**** | ****Bit 6**** | ****Bit 5**** | ****Bit 4**** | ****Bit 3**** | ****Bit 2**** | ****Bit 1**** | ****Bit 0**** |
| **0** | (Padding) | | | Kana | Compose | Scroll Lock | Caps Lock | Num Lock |

## Consumer Control and Power TLC

The following report descriptor represents the Consumer Control TLC and the Power TLC.

Usage Page (Consumer)

Usage (Consumer Control)

Collection (Application)

Report ID (2)

Logical Minimum (0x00)

Logical Maximum (0x3FF)

Usage Minimum (0x00)

Usage Maximum (0x3FF)

Report Size (16)

Report Count (1)

Input (Data, Array, Absolute)

End Collection

;

Usage Page (Generic Desktop)

Usage (System Control)

Collection (Application)

Report ID (3)

Usage Minimum (System Sleep)

Report Size (1)

Report Count (1)

Input (Data, Variable, Relative, Preferred)

Report Count (7)

Input (Constant)

End Collection

This HID interface issues one of two possible input reports, distinguished by Report IDs:

* **Consumer control—**Report ID (2) is used to communicate CCBs:
  + When a user presses a key, an input report is generated to identify the key. When the key is released, an input report with usage value=0 is issued.
  + Only one usage is active and sent at a time. Consumer controls do not allow multiple buttons to be pressed simultaneously. When a new usage is sent, it is assumed that the usage for the previous key is released.
  + Some keys are retriggering keys (such as volume or brightness increase) and their rate of repeat is handled by Windows. Hardware should not keep resending the usage when these keys are kept depressed by user. Hardware should only send an input report when a button is pressed down and another when it is released.
* **System Power control—**Report ID (3) is used to communicate power buttons only:
  + When the user presses and releases the Sleep key, the input report should be generated when the key is pressed. A following input report should be generated when the key is released.

Table 11. Consumer control input report

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ****Byte**** | ****Bit 7**** | ****Bit 6**** | ****Bit 5**** | ****Bit 4**** | ****Bit 3**** | ****Bit 2**** | ****Bit 1**** | ****Bit 0**** |
| **0** | Report ID = 2 | | | | | | | |
| **1** | Usage value | | | | | | | |
| **2** |

Table 12. System power control input report

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ****Byte**** | ****Bit 7**** | ****Bit 6**** | ****Bit 5**** | ****Bit 4**** | ****Bit 3**** | ****Bit 2**** | ****Bit 1**** | ****Bit 0**** |
| **0** | Report ID = 3 | | | | | | | |
| **1** | (padding) | | | | | | | Sleep |

# Appendix 2: AppCommands Remapping

A number of AppCommands have been defined in Windows 8. A list of them can be found in Winuser.h. These AppCommands are associated with special registry keys, and each key defines an application associated with a particular AppCommand. The keys are stored in the following locations:

HKLM\ Software\ Microsoft\ Windows\ CurrentVersion\ Explorer\ AppKey \<*NUM*>  
HKCU\ Software\ Microsoft\ Windows\ CurrentVersion\ Explorer\ AppKey \<*NUM*>

Replace <NUM> with the appropriate numeric value for the AppCommands as defined in Winuser.h.

The value name stored inside each key should correspond to one of the values in the following table.

Table 13. AppCommand remapping types

|  |  |  |
| --- | --- | --- |
| Type | Definitions | Example |
| ***ShellExecute*** | Used to define a particular application that is launched when this key is pressed. Arguments can also be passed (enclosed within the quotes). | -Calc.exe -“runme.exe / arg” |
| ***Association*** | Establishes a relationship between a file extension and a program to be launched for that particular extension. | .mp3 |
| ***RegisteredApp*** | Used to launch an application that is registered with the operating system. | mail |

The data stored in the registry value field must be in the form of a string (REG\_SZ). The registry keys are first searched in the current user hive of the registry. If a match is not found in this section, the local machine section is then searched. Ensure that the registry keys do not contain incorrect values that can cause a key to become useless.

Changing the value stored inside a registry key does not require a reboot. These keys contain the default values associated with the corresponding AppCommand and must not be removed.

Vendors can write their own user interface to program the registry keys or use INF files to install special entries as defaults.

# Appendix 3: Implementing Airplane Mode Radio Management Switch/Button/LED

Windows 8 provides users with the convenience of switching wireless radios on and off (as with Airplane mode) using UI or a physical switch or button. The following sections provide an overview about radio switch support and the two types of solutions available to OEMs and PC manufacturers:

* **Software Solution**: Windows 8 enables PC manufacturers to provide a vendor-specific HID minidriver to identify the status of the switch.
* **Hardware Solution**: Windows 8 enables PC manufacturers to build their next-generation hardware leveraging existing HID transports, alleviating the need for OEM-specific drivers.

The following software stack diagram summarizes how the hardware and drivers interface with the operating system. Note that the only differences between the software and hardware solutions are as follows:

1. Is the HID miniport specific to Airplane mode only (software solution) or to an industry standard (hardware solution)?
2. Is the KMDF HID miniport written by PC manufacturers (software solution) or is it an industry standard HID transport solution (hardware solution)?



Figure 2: Software Stack Diagram

## Software Solution (for Existing Systems)

A software solution is primarily meant for PC manufacturers who have existing PCs and want to make their current hardware work with Windows 8 Airplane mode implementation. The solution requires the PC manufacturer to develop, sign, distribute, and maintain a KMDF HID miniport driver that translates the status of their ACPI Airplane mode switch and convert it into a standard HID report that is expected by Windows 8.

### Guidance for PC Manufacturers

PC manufacturers can leverage the [Radio Switch Test Driver for OSR USB-FX2 Development Board](http://code.msdn.microsoft.com/windowshardware/Radio-Switch-Test-Driver-0eb8f876) to develop an OEM-specific HID-compliant KMDF miniport driver that communicates to the OEM proprietary interface (such as an embedded controller) on the bottom edge and Hidclass on the upper edge. This driver identifies the hardware wireless switch to Windows. OEMs can also use the [Radio Switch Test Driver for OSR USB-FX2 Development Board](http://code.msdn.microsoft.com/windowshardware/Radio-Switch-Test-Driver-0eb8f876) sample. The Radio Management API and HID-compliant driver communicate via HID reports (messages). The format of the HID report is defined in each device in a HID report descriptor.

When the Radio Management API sends a HID message, the switch must consume the message and toggle the state because the global wireless state has changed. In the same way, when the state of the hardware wireless switch changes, the HID-compliant hardware switch/button must expose the HID collections to be consumed by the Radio Management API (for more information, see the Wireless Radio Control section earlier in this paper). The wireless LED must use the HID-compliant driver to reflect the state of the Airplane mode switch located in the user interface. The wireless LED uses HID only for output (there is no input because there is no button). For more information about the Radio Management API, see [Radio Management](http://msdn.microsoft.com/en-us/library/hh406615(v=VS.85).aspx).

**Note**: If you are developing a HID driver for an A-B switch, your proprietary embedded controller must report the correct state of the switch at all times by sending a HID message to the HID driver, including every time the PC is turned on back on. Reporting the state of the A-B switch when the computer is turned back on is especially important in the case that the switch changed states while the PC was in states S3/S4/S5.

### Best Practices for Third-Party Drivers

The driver must support the following three key operational IOCTLS for HID Switch integration:

1. IOCTL\_HID\_GET\_INPUT\_REPORT
2. IOCTL\_HID\_SET\_OUTPUT\_REPORT
3. IOCTL\_HID\_READ\_REPORT

In the Radio Driver sample, the driver response to these IOCTLs can be seen in the **HidFx2EvtInternalDeviceControl( )** function in hid.c.

This is the KMDF driver **EvtIoInternalDeviceControl** event callback function.

Here each of the IOCTLs is handled by a separate case in the switch statement.

**IOCTL\_HID\_GET\_INPUT\_REPORT**

* Typically, this IOCTL is called only if the radio switch is a slider type.
* When the Radio Management API needs to know the current switch state, it calls **HidD\_GetInputReport()**, which results in this IOCTL at the driver.
* The driver should complete this IOCTL immediately and return the current state of the switch in the framework request object.

**IOCTL\_HID\_SET\_OUTPUT\_REPORT**

* A driver sees this IOCTL only if it reported a HID descriptor for LED.
* When the Radio Management API needs to set the current LED state, it calls **HidD\_SetOutputReport()**, which results in this IOCTL at the driver.
* The driver should complete this IOCTL immediately and set the LED state based on the data passed in the framework request object.

**IOCTL\_HID\_READ\_REPORT**

* As part of the ShellHWDetection service, a worker thread runs whenever a Radio Management HID device is installed. The worker thread blocks on a read of the HID device, and the read waits until either the service shuts down or a HID report is received. Once the HID report is processed, it loops back to the blocking read.
* At the driver level, the read results in an IOCTL\_HID\_READ\_REPORT being pended to the device. This IOCTL should not be completed immediately, but rather the request should be queued until the driver needs to report a change in the switch state.
* In the [Radio Switch Test Driver for OSR USB-FX2 Development Board](http://code.msdn.microsoft.com/windowshardware/Radio-Switch-Test-Driver-0eb8f876) sample code, the switch statement for this IOCTL uses a call to **WdfRequestForwardToIoQueue()** to re-queue the I/O request.
* In the sample code, when a switch changes state, there is de-bouncing of the input, which finally results in a timer function being called (**HidFx2EvtTimerFunction** in usb.c). This function uses a call to **WdfIoQueueRetrieveNextRequest()** to retrieve the queued I/O request. It then completes the I/O and returns the current state of the switch.
* Once the I/O is completed and the service has read the new switch state, it loops again and a new IOCTL\_HID\_READ\_REPORT is presented to the driver.

**Note**: Whenever a driver is in the D0 state, it has an outstanding IOCTL\_HID\_READ\_REPORT I/O request. This I/O request is cancelled and re-issued when the device goes through a D0->Dx->D0 transition. As soon as the system wakes up from sleep/hibernate/shutdown, the state of the switch should be reported via a HID message.

The following table summarizes the steps for successfully using a software solution.

Table 14. Steps for a successful software solution

|  |  |  |
| --- | --- | --- |
| Steps to success | Work summary | Success metric |
| 1 | Update BIOS   * Expose switch devnode via ACPI. * Expose LED (if exists) via same ACPI. * Don’t control radios via BIOS when the switch/button is toggled. | * A devnode is created and visible in the Device Manager for optional switch/button/LED. Note that devnode appears as “Unknown Device” in the Device Manager until the OEM’s INF matches this Hardware ID. * Ensure that ACPI reports a correctly formatted Hardware ID that is unique to the make/model of the system. |
| 2 | Create a KMDF driver (starting with WDK sample) that loads on devnode above. | HID miniport driver loads on the Hardware ID generated by ACPI. |
| 3 | Update driver to talk to BIOS/EC and retrieve current hardware wireless switch settings. | * HID driver reports hardware switch setting accurately. * If the HID driver works correctly, a flyout indicating “Airplane Mode On” appears for a few seconds when the hardware switch is turned off (all wireless off). |
| 4 | Update driver to talk to BIOS/EC and set current Wireless LED settings (if optional LED is present). | * HID driver passes software-defined LED status to LED and accurately turns it on and off. * If the HID driver works correctly, the wireless LED turns off when the hardware switch is turned off (all wireless off). |
| 5 | Validate that the hardware wireless switch/button solution passes the System.Client.RadioManagement HCK requirements and tests. | * Pass all appropriate Windows 8 Certification tests. * Certification tests prove correct functionality of sending and receiving on/off messages using HID messages. |

### Installation Scenarios

OEM Factory Install

The PC manufacturer’s third-party Airplane mode HID miniport driver (for their hardware wireless switch) must be pre-installed. It is also a best practice to populate this driver on Windows Update for users or IT firms that need to rebuild the system image. These drivers will not be included with Windows 8.

#### Upgrade Scenarios

The PC manufacturer’s third-party Airplane mode HID miniport driver (for their hardware wireless switch) must be uploaded in Windows Update so that users have full functionality of the hardware wireless switch in upgraded PCs. The user may also need the latest ACPI BIOS (or related drivers) installed to expose the correct PDO to load this driver on.

For more information about the third- party driver for the wireless switch, please contact [rmfc@microsoft.com](mailto:rmfc@microsoft.com).

## Hardware Solution (for New Systems)

PC manufacturers who are building new systems are encouraged to report the Airplane mode radio management switch leveraging Windows 8 inbox supported HID Transports. These transports currently include USB, Bluetooth and I2C (especially recommended for SoC formfactor systems as compared to solution #1).

PC manufacturers can use the approach below to either create a slider switch solution or leverage the same steps to create a solution on their HID keyboards (USB or I2C). Do note that the airplane mode solution only works on mobile formfactor PCs; hence a user with a USB keyboard will be able to set their laptop into airplane mode but not their desktop.

The following table summarizes the steps for successfully using a hardware solution.

Table 15. Steps for a successful hardware solution

|  |  |  |
| --- | --- | --- |
| Steps to success | Work summary | Success metric |
| 1 | Expose correct ACPI entries correctly enumerate your device over USB or I2C. | * A devnode is created and visible in the Device Manager for optional switch/button/LED. * The appropriate transport driver (hidusb.sys or hidi2c.sys) is loaded and visible in Device Manager. |
| 2 | Send the correct Report Descriptor (see samples below for details). | HID class driver enumerates the correct PDO for this device and Windows 8 recognizes it. |
| 3 | Validate that the hardware wireless switch/button solution passes the System.Client.RadioManagement WHCK requirements and tests. | * Pass all appropriate Windows 8 Certification tests. * Certification tests prove correct functionality of sending and receiving on/off messages using HID messages. |

For more information about leveraging HID I2C for this solution, please contact [hidusbfb@microsoft.com](mailto:hidusbfb@microsoft.com).

## Sample Report Descriptors

HID Review Request 40 standardized HID usage IDs for radio management, on the generic desktop usage page, as follows:

Table 16. HID usage IDs (on generic desktop usage page) for radio management

|  |  |  |
| --- | --- | --- |
| ****Usage ID**** | ****Usage name**** | ****Usage type**** |
| 0x000C | Wireless Radio Controls | Collection Application (CA) |
| 0x00C6 | Wireless Radio Button | On/Off Control (OOC) |
| 0x00C7 | Wireless Radio LED | On/Off Control (OOC) |
| 0x00C8 | Wireless Radio Slider Switch | On/Off Control (OOC) |

Devices that support Airplane mode must use the following top-level collection:

* USAGE\_PAGE (Generic Desktop)                    05 01
* USAGE (Wireless Radio Controls)                   09 0C

The following section provides sample report descriptors that PC manufacturers must leverage. Please note that if the top-level collection is part of a report descriptor that already has another top-level collection, a Report ID must be included.

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

USAGE\_PAGE (Generic Desktop) 05 01

USAGE (Wireless Radio Controls) 09 0C

COLLECTION (Application) A1 01

LOGICAL\_MINIMUM (0) 15 00

LOGICAL\_MAXIMUM (1) 25 01

USAGE (Wireless Radio Button) 09 C6

REPORT\_COUNT (1) 95 01

REPORT\_SIZE (1) 75 01

INPUT (Data,Var,Rel) 81 06

REPORT\_SIZE (7) 75 07

INPUT (Cnst,Var,Abs) 81 03

END\_COLLECTION C0

**Button with LED – for laptops, tablets, and convertibles that do not support connected standby**

USAGE\_PAGE (Generic Desktop) 05 01

USAGE (Wireless Radio Controls) 09 0C

COLLECTION (Application) A1 01

LOGICAL\_MINIMUM (0) 15 00

LOGICAL\_MAXIMUM (1) 25 01

USAGE (Wireless Radio Button) 09 C6

REPORT\_COUNT (1) 95 01

REPORT\_SIZE (1) 75 01

INPUT (Data,Var,Rel) 81 06

REPORT\_SIZE (7) 75 07

INPUT (Cnst,Var,Abs) 81 03

USAGE (Wireless Radio LED) 09 C7

REPORT\_SIZE (1) 75 01

OUTPUT (Data,Var,Rel) 91 02

REPORT\_SIZE (7) 75 07

OUTPUT (Cnst,Var,Abs) 91 03

END\_COLLECTION C0

**Note**: When using a radio manager BUTTON, the PC manufacturer should send one HID report when the button is released and not when the button is pressed. This is because the Toggle button is generally a relative input and not absolute.

**Slider switch (without LED) – for laptops, tablets, and convertibles**

USAGE\_PAGE (Generic Desktop)                    05 01   
USAGE (Wireless Radio Controls)                 09 0C   
COLLECTION (Application)                        A1 01   
LOGICAL\_MINIMUM (0)                             15 00   
LOGICAL\_MAXIMUM (1)                             25 01   
USAGE (Wireless Radio Slider Switch)           09 C8   
REPORT\_COUNT (1)                                95 01   
REPORT\_SIZE (1)                                 75 01   
INPUT (Data,Var,Abs)                            81 02   
REPORT\_SIZE (7)                                 75 07   
INPUT (Cnst,Var,Abs)                            81 03   
END\_COLLECTION                                  C0

**Slider switch with LED** **– for laptops, tablets, and convertibles that do not support connected standby**

USAGE\_PAGE (Generic Desktop) 05 01

USAGE (Wireless Radio Controls) 09 0C

COLLECTION (Application) A1 01

LOGICAL\_MINIMUM (0) 15 00

LOGICAL\_MAXIMUM (1) 25 01

USAGE (Wireless Radio Slider Switch) 09 C8

REPORT\_COUNT (1) 95 01

REPORT\_SIZE (1) 75 01

INPUT (Data,Var,Abs) 81 02

REPORT\_SIZE (7) 75 07

INPUT (Cnst,Var,Abs) 81 03

USAGE (Wireless Radio LED) 09 C7

REPORT\_SIZE (1) 75 01

OUTPUT (Data,Var,Rel) 91 02

REPORT\_SIZE (7) 75 07

OUTPUT (Cnst,Var,Abs) 91 03

END\_COLLECTION C0

**LED only (no button or slider) – for laptops, tablets, and convertibles that do not support connected standby**

USAGE\_PAGE (Generic Desktop) 05 01

USAGE (Wireless Radio Controls) 09 0C

COLLECTION (Application) A1 01

LOGICAL\_MINIMUM (0) 15 00

LOGICAL\_MAXIMUM (1) 25 01

USAGE (Wireless Radio LED) 09 C7

REPORT\_COUNT (1) 95 01

REPORT\_SIZE (1) 75 01

OUTPUT (Data,Var,Rel) 91 02

REPORT\_SIZE (7) 75 07

OUTPUT (Cnst,Var,Abs) 91 03

END\_COLLECTION C0

## Validating Airplane Mode

Windows 8 operating system (EEAP build 8302+) provides support for this functionality. Requirements and tests are also available in the [Windows Hardware Certification Kit](http://msdn.microsoft.com/library/windows/hardware/hh748188) to validate the functionality of your solution for the wireless switch.

* WHCK requirement: System.Client.RadioManagement.RadioManagementAPIHID
* WHCK test tools: HID Radio Button - Verify HID Usages

For more information about the third-party driver for the wireless switch, please contact [rmfc@microsoft.com](mailto:rmfc@microsoft.com).