Using the Universal Driver for Windows SideShow

Includes the Protocol for Windows SideShow Specification

December 19, 2008

Abstract

This paper provides information about the universal driver and the standard wire protocol for the Windows® SideShow™ feature of the Windows family of operating systems. It provides guidelines for device developers who want to use the universal driver to connect certain kinds of SideShow-compatible devices to computers that are running Windows.

This information applies for the following desktop operating systems:  
 Windows Server® 2008 (with the Desktop Experience Pack)  
 Windows Vista®

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

For the latest information, see:   
 http://www.microsoft.com/whdc/device/sideshow/Univ-Drv\_Sideshow.mspx

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

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

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

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

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

© 2008 Microsoft Corporation. All rights reserved.

Microsoft, MSDN, Windows, Windows Server, SideShow, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

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

Document History

| **Date** | **Change** |  |  |  |
| --- | --- | --- | --- | --- |
| December 19, 2008 | New “Reconnection Guidelines” under “Bluetooth,” page 7.  Various Remarks updates in “Protocol for Windows SideShow.” | | | |
| May 16, 2008 | First publication | | | |

Contents

[Overview 4](#_Toc217379288)

[SideShow-Compatible Device Options 4](#_Toc217379289)

[About the Universal Driver 5](#_Toc217379290)

[Transports 5](#_Toc217379291)

[USB Transport 5](#_Toc217379292)

[Hardware ID Requirements 5](#_Toc217379293)

[Microsoft OS Feature Descriptor Requirements 5](#_Toc217379294)

[USB Device Descriptor Requirements 6](#_Toc217379295)

[Interfaces and Endpoints 6](#_Toc217379296)

[Decoding Data 7](#_Toc217379297)

[Bluetooth Transport 7](#_Toc217379298)

[Establishing a Connection with a PC 7](#_Toc217379299)

[Reconnection Guidelines 7](#_Toc217379300)

[Custom Bluetooth Service Requirements 8](#_Toc217379301)

[TCP/IP Transport 9](#_Toc217379302)

[PnP-X Implementation 9](#_Toc217379303)

[Establishing a Connection with a PC 9](#_Toc217379304)

[Connection Requirements 9](#_Toc217379305)

[SSL Communications 10](#_Toc217379306)

[Custom I/O Control Codes 10](#_Toc217379307)

[Device Restart 10](#_Toc217379308)

[Passthrough 11](#_Toc217379309)

[Custom Icons (Optional) 11](#_Toc217379310)

[Protocol for Windows SideShow 12](#_Toc217379311)

[Overview 12](#_Toc217379312)

[Packet Header 12](#_Toc217379313)

[Packet Types 13](#_Toc217379314)

[Response Packets 13](#_Toc217379315)

[ACK 13](#_Toc217379316)

[NAK 13](#_Toc217379317)

[Command Packets 14](#_Toc217379318)

[Ping 14](#_Toc217379319)

[SendPassThrough 14](#_Toc217379320)

[Reset 15](#_Toc217379321)

[SetUserState 15](#_Toc217379322)

[SetCurrentUser 16](#_Toc217379323)

[GetCurrentUser 16](#_Toc217379324)

[GetDeviceFirmwareVersion 17](#_Toc217379325)

[GetCapabilities 17](#_Toc217379326)

[GetApplicationOrder 18](#_Toc217379327)

[SetApplicationOrder 19](#_Toc217379328)

[SetLanguage 19](#_Toc217379329)

[GetPreEnabledApplications 20](#_Toc217379330)

[SetTime 21](#_Toc217379331)

[SetShortDateFormat 21](#_Toc217379332)

[SetLongDateFormat 22](#_Toc217379333)

[SetShortTimeFormat 22](#_Toc217379334)

[SetLongTimeFormat 23](#_Toc217379335)

[AddApplication 23](#_Toc217379336)

[DeleteApplication 24](#_Toc217379337)

[DeleteAllApplications 25](#_Toc217379338)

[AddNotification 25](#_Toc217379339)

[DeleteNotification 26](#_Toc217379340)

[DeleteAllNotifications 26](#_Toc217379341)

[SetNotificationsEnabled 26](#_Toc217379342)

[AddContentItem 27](#_Toc217379343)

[DeleteContentItem 28](#_Toc217379344)

[DeleteAllContentItems 28](#_Toc217379345)

[GetSupportedEndpoints 29](#_Toc217379346)

[SetTimeZone 29](#_Toc217379347)

[GetDeviceName 30](#_Toc217379348)

[GetDeviceManufacturer 30](#_Toc217379349)

[Sync 31](#_Toc217379350)

[Event Packets 32](#_Toc217379351)

[ContentMissing 32](#_Toc217379352)

[ApplicationEvent 32](#_Toc217379353)

[ChangeUserRequestEvent 33](#_Toc217379354)

[Resources 34](#_Toc217379355)

# Overview

Microsoft provides a common driver for devices that are compatible with Windows® SideShow™. This driver, called the *universal driver,* can significantly reduce how long it takes to create and test a SideShow-compatible device because Microsoft provides the device driver.

This paper provides the information that you need to use the universal driver to enable your device to provide SideShow functionality.

# SideShow-Compatible Device Options

You have several options for implementing a device driver for connecting your SideShow-compatible device to a PC that is running Windows. The following table describes these options, in the order of increasing development complexity.

| Device implementation | Driver type | Description |
| --- | --- | --- |
| .NET Micro Framework-based | Universal driver | Devices that are based on the Windows SideShow Device Software Development Kit (SDK) for .NET Micro Framework require no driver work. These devices simply use the universal driver. No additional work is required. |
| Windows CE-based (offline) | Universal driver | SideShow devices that are Windows CE–based and have offline support (that is, they allow interact ion with the gadget when the PC is out-of-range or off) simply use the universal driver. |
| Windows CE-based (online-only) | Bitmap driver | SideShow devices that are Windows CE–based and are online only (that is, the devices require the computer to be on for the device to be operational) can use the Windows SideShow Bitmap Driver instead of the universal driver. For information about the SideShow Bitmap Driver, see <http://www.codeplex.com/SideShowBitmap>. |
| Other implementations (online-only) | Bitmap driver | Online-only SideShow devices (that is, the devices require the computer to be on for the device to be operational) can use the Windows SideShow Bitmap Driver. For information about the SideShow Bitmap Driver see <http://www.codeplex.com/SideShowBitmap>. |
| Other implementations | Universal driver | SideShow devices that are based on implementations such as Linux can use the universal driver to connect to PCs that are running Windows. This paper describes how to use the universal driver and standard wire protocol for this option. |
| Other implementations | Custom driver | To implement a custom solution that includes a custom device driver, use the Windows Driver Kit (WDK) instead of this paper. |

# About the Universal Driver

The universal driver for SideShow, which is a user-mode driver that is built against the [Windows user-mode driver framework (UMDF)](http://msdn2.microsoft.com/en-us/library/aa511018.aspx), uses the SideShow class extension component. The universal driver standardizes how SideShow-enabled devices connect to computers that are running Windows. The universal driver provides the following features:

* Support for various transports. The universal driver can connect to devices by using USB, Bluetooth, or TCP/IP.
* Custom input/output control codes (IOCTLs). The universal driver supports two new features that extend the functionality of the class extension component: device restart and data passthrough.
* Custom icons. Device manufacturers can provide a custom graphic that Windows uses to represent the device as part of the SideShow Control Panel user interface.
* A standard wire protocol. The universal driver uses the Protocol for Windows SideShow for communications with the device, such as issuing commands, receiving responses, and receiving events.

For detailed information about how SideShow drivers work, see “Windows SideShow” in the WDK.

The following sections provide details about the universal driver features, including the Protocol for Windows SideShow specification.

# Transports

The following sections provide information about how to connect to the universal driver by using USB, Bluetooth, or TCP/IP.

## USB Transport

This section provides details about how to use the USB transport to communicate with the universal driver.

### Hardware ID Requirements

Hardware manufacturers must provide a unique hardware ID for each device model. Typically, the hardware ID is created by combining a vendor ID (VID) with a product ID (PID). The VID must be assigned by the USB Consortium. The PID must be unique among devices that are manufactured by the vendor and must be associated with one device model.

When testing your device, you can use any combination of VID and PID.

### Microsoft OS Feature Descriptor Requirements

USB-connected devices that use the universal driver to connect to PCs that are running Windows must support Microsoft OS descriptors. To learn about Microsoft OS descriptors, see “Microsoft OS Descriptors” on the WHDC Web site.

When a user connects a USB device to a computer, Windows compares identifiers in the firmware with the available INF files to determine the most appropriate driver for the device. These INF files can be on the system, on vendor media, on Windows update, and so on.

To enable Windows to match the universal driver INF file to your device, you can store a Microsoft OS feature descriptor in the device’s firmware. The extended compatible ID OS feature descriptor enables you to store one or more compatible IDs and subcompatible IDs in the device firmware on a per-interface or per-function basis. To associate the universal driver with your device, you must store the following extended compatible ID values:

* **Compatible ID:** SIDESHW
* **Subcompatible ID:** UNIV1

The universal driver INF file contains the following representation of this ID:

USB\MS\_COMP\_SIDESHW&MS\_SUBCOMP\_UNIV1

#### USB Version Requirements

To use the Microsoft OS feature descriptor functionality, the device must support USB 2.0. Because a USB2.0 full-speed device is functionally equivalent to a USB1.1 device, any USB1.1 device can use a USB version descriptor of 0x0200 and can support the required feature descriptor.

### USB Device Descriptor Requirements

You are not required to specify a USB class or subclass. The *Universal Serial Bus Revision 2.0 Specification* does not define categories for SideShow devices.

You must provide a unique serial number for each device unit that is manufactured.

The device manufacturer must configure the manufacturer and product name descriptors. The universal driver displays these string descriptors in the Windows user interface. Therefore, these strings must contain accurate, descriptive information that uniquely identifies each device model.

To enable maximum power savings for PCs that are running on battery power, the device must support the USB Selective Suspend feature.

### Interfaces and Endpoints

The device must expose a single interface, which must support the following two bulk endpoints:

* **BULK IN**. The recommended endpoint address is 0x81.
* **BULK OUT**. The recommended endpoint address is 0x02.

The endpoint attributes must specify only the endpoint type (bulk). The device determines the maximum packet size. We recommend that the maximum packet size be as large as possible.

### Decoding Data

Data must be provided as a raw bit stream to the upper-layer application. The application parses the stream into appropriate packets and decodes the data, as documented in “Protocol for Windows SideShow” later in this paper. The USB driver must pass the data stream unchanged to applications.

## Bluetooth Transport

This section provides details about how to use the Bluetooth transport to communicate with the universal driver. (Note that the Bluetooth transport is not supported on Windows Server® 2008.)

### Establishing a Connection with a PC

Windows Vista® enables users to set Bluetooth connection properties through Control Panel. To appear in the list of available devices, a Bluetooth enabled device must broadcast its availability for connection. This broadcast is called *discoverability*. Typically, such devices provide user interface elements that enable users to turn on or turn off discoverability and to provide a security password code, if it is required. To enable users to change Bluetooth settings on the device, we recommend that you use the device operating system’s default Bluetooth settings features and user interface, if this is possible.

The following steps describe, generally, how a Bluetooth enabled device connects to the universal driver:

1. The device publishes a Service Discovery Protocol (SDP) record that contains the custom service ID and specifies the required protocols.

2. The user pairs the PC with the Bluetooth enabled device.

3. Windows loads the universal driver.

4. During initialization, the driver publishes a custom SideShow SDP record that contains the Bluetooth MAC address for the device and the channel for the device to use to initiate the connection to the PC.

5. The driver waits for the device to connect.

6. The device connects to the PC, if it is required.

7. The device ends the connection, when it is finished.

Because the universal driver acts as a server, the device can open and close the connection as required. This behavior conserves battery power by using the Bluetooth enabled radio only when it is necessary.

### Reconnection Guidelines

A device that continually attempts to reconnect to the PC can drain the device’s battery or interfere with other communications that use the Bluetooth enabled radio. One way to prevent unnecessary reconnections is for the device to attempt reconnection only on device events such as waking from an idle state, system boot or restart, or user interaction with the device.

The device must initiate a reconnection, and the events that trigger device reconnection depend on the usage scenario. The reconnection behavior for a device that is assumed to be in the same room or building as the PC, such as a remote control, are different from the behavior of a device that is not always in range of the PC, such as a mobile phone. For instance, as soon as a remote control wakes from an idle state, it should attempt to reconnect to the PC. A remote control may also initiate reconnection when it is removed from its cradle or when an accelerometer inside the remote control indicates that the remote control has been moved or lifted. The remote control can disconnect to save battery life when it has not been moved for some time.

You must limit the number of consecutive reconnection attempts, especially for a device that is not always in range of the PC. For example, a phone should provide a user interface for the user to manually initiate a connection to the PC. After the user chooses to connect, the device should attempt to connect no more than the specified number of times.

### Custom Bluetooth Service Requirements

This section describes the requirements for communicating with the universal driver custom Bluetooth service.

#### Communication Protocol

The Bluetooth enabled device must support the Bluetooth Radio Frequency Communications (RFCOMM) protocol. The device must use Bluetooth RFCOMM directly to communicate with the PC.

#### SDP Record Published by the Device

The SDP record must be published before the user initiates the Bluetooth connection from the PC.

Because Bluetooth RFCOMM uses the Logical Link Control and Adaptation Protocol (L2CAP), the device must specify Bluetooth RFCOMM and L2CAP in the **ProtocolDescriptorsList** attribute.

The SDP record must contain the following custom service ID:

{5d9dda39-1e82-49c7-a0d6-6507ba9287ef}

#### SDP Record Published by the Universal Driver

The universal driver publishes a Bluetooth service ID that differs from that of the device. The ID that the driver publishes is unique for each connected device. The ID contains a hint that you must use to determine which SDP record is associated with your device. Service IDs are defined as globally unique identifiers (GUIDs).The following struct describes the **GUID** data type:

typedef struct \_GUID

{

    DWORD   Data1; // 4 bytes

    WORD    Data2 ;  // 2 bytes

    WORD    Data3;  // 2 bytes

    BYTE    Data4[8]; // 8 bytes total

} GUID;

The universal driver encodes the Bluetooth MAC address as the first 6 bytes (in little-endian order) of the 8‑byte **Data4** member of the service ID GUID. For example, assume that a hypothetical device paired with a PC that is running Windows uses the following Bluetooth MAC address:

0x00081b00881b

The custom service ID GUID that is published by the universal driver that is associated with this Bluetooth enabled device is the following:

5d9dda39-1e82-49c7-**1B88-001B0800**87ef

The format of the service ID GUID in memory appears as follows:

39 da 9d 5d 82 1e c7 49 **1b 88 00 1b 08 00** 87 ef

Your device’s connection implementation must compare these 6  bytes (marked as **bold** text in the previous examples) with the device’s Bluetooth MAC address to determine which particular SDP record describes the current driver connection for the device.

## TCP/IP Transport

This section provides details about how to use the TCP/IP transport to communicate with the universal driver.

### PnP-X Implementation

Plug and Play Extensions (PNP‑X) for Windows specifies a set of extensions to Plug and Play that deliver support for network-connected devices. PnP-X enables network-connected devices to appear as devices in the Windows operating system and provides an installation experience that is similar to attaching a physically connected device to a personal computer. The *Plug and Play Extensions for Windows Specification* describes these extensions and provides requirements and guidelines for hardware manufacturers to create devices that can be installed by using PnP-X. To use TCP/IP to communicate with the universal driver, you must follow the guidelines in “PnP‑X: Plug and Play Extensions for Windows Specification” on the WHDC Web site.

### Establishing a Connection with a PC

For users to connect to a device on the network, the device must be discoverable and Windows Vista network discovery must be enabled. Users must use Network Explorer to install the device, as described in the PnP‑X specification.

For TCP/IP connections, the device acts as the server and the driver acts as the client. When PnP-X initializes the universal driver, the universal driver tries to connect to the device. The universal driver repeatedly tries to make a connection, until either the connection is established or the connection handshake fails.

### Connection Requirements

The device controls when connections can occur. The device must broadcast its presence when it is ready to accept a connection from the PC that is running Windows. To end the connection, the device must stop broadcasting its presence.

#### Communication Protocols

The universal driver supports both IPv4 and IPv6 connections. The driver initiates a single connection to the device, based on the protocol that the PC that is running Windows used to recognize the device.

#### Hardware ID

We recommend that the device hardware ID be unique for each device model. However, the universal driver uses PnP-X compatible IDs to identify devices.

#### Compatible IDs

PnP-X uses compatible identifiers to identify devices and locate compatible drivers. To use the universal driver, the device must set its compatible ID to one of the values in the following table.

| **Compatible ID** | **Transport** | **Port number** |
| --- | --- | --- |
| PnPX\_WindowsSideShow\_CPID | Standard TCP/IP socket | 5360 |
| PnPX\_WindowsSideShow\_SSL\_CPID | SSL-encrypted socket | 5361 |

#### Other Metadata

We recommend that you configure properties such as friendly name, manufacturer, model, and device URL by using appropriate values for the device.

#### Category

The device must specify the following full category name:

Displays.WindowsSideShow

### SSL Communications

If the network device specified the ID that is compatible with a secure sockets layer (SSL), the universal driver tries to create an SSL-encrypted connection to the device by using the SSL/TLS (Transport Layer Security) standard. The device may provide either a self-signed X.509 certificate that contains a public key or a valid trusted root signed certificate. The device must use a certificate that is unique for each device unit. We recommend that this certificate be generated on the device’s first startup. The universal driver does not use the certificate to authenticate the device, but the certificate is used to validate that the device is the one that the user has chosen to install and to encrypt the communication channel. Each device must use the same unique certificate every time that it connects to the PC that is running Windows.

# Custom I/O Control Codes

The universal driver recognizes two custom IOCTLs, which are described in the following sections.

### Device Restart

Restarts the device. The IOCTL is defined as follows:

**Constant:** IOCTL\_ SIDESHOW\_REBOOT\_DEVICE

**Device type:** 0x8010

**Function code:** 0x810

**Transfer type:** METHOD\_BUFFERED

**Required access:** FILE\_WRITE\_DATA

**Input parameter:** A single input parameter of type **DWORD**. The high 4 bits are reserved for optional vendor-specific values. The following table describes possible values.

| **Value** | **Description** |
| --- | --- |
| 0 | Restart. |
| 1 | Restart and wait in bootloader. This enables firmware upgrade, for some devices. |

**Output parameter:** A return code of type **HRESULT**. The response can contain one of the HRESULTS in the following table.

| **HRESULT** | **Value** | **Description** |
| --- | --- | --- |
| S\_OK | 0x00000000 | Acknowledged and acted on (ACK). |
| E\_NOTIMPL | 0x80004001 | Not acknowledged (NAK). |

###### Remarks

After the universal driver receives IOCTL\_ SIDESHOW\_REBOOT\_DEVICE, the driver sends the universal wire protocol **Reset** command. See **Reset** in the “Command Packets” section of “Protocol for Windows SideShow” later in this paper.

### Passthrough

Provides direct, or *passthrough*, communication with the device:

**Constant:** IOCTL\_SIDESHOW\_ PASSTHROUGH

**Device type:** 0x8010

**Function code:** 0x811

**Transfer type:** METHOD\_BUFFERED

**Required access:** FILE\_READ\_DATA|FILE\_WRITE\_DATA

###### Remarks

After the universal driver receives IOCTL\_ SIDESHOW\_ PASSTHROUGH, the driver sends the universal wire protocol **SendPassThrough** command. For more information about this command packet, see **SendPassThrough** in the “Command Packets” section of “Protocol for Windows SideShow” later in this paper.

# Custom Icons (Optional)

The universal driver can retrieve a small graphics file from each device to display as an icon in Windows Device Manager and Control Panel. The icon is provided as a standard ICO file that has the following properties:

* The icon must be provided in a minimum of three sizes: 48x48, 32x32, and 16x16 pixels.
* Each icon must contain at least the following three color depths:
* 24-bit with 8-bit alpha (32-bit)
* 8-bit (256 colors) with 1-bit transparency
* 4-bit (16 colors) with 1-bit transparency

The driver retrieves the icon from the device by issuing the following SideShow device capability query:

DEFINE\_PROPERTYKEY(SIDESHOW\_CAPABILITY\_CUSTOM\_DEVICE\_ICON,  0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 17); // [VT\_UI1 | VT\_VECTOR]

For information about how the universal driver sends a device capability query, see **GetCapabilities** in the “Command Packets” section of “Protocol for Windows SideShow” later in this paper.

The device returns the ICO file as a byte array in an encoded variant of type VT\_UI1 | VT\_VECTOR. If it does not provide a custom icon, the device should return an empty variant (VT\_EMPTY).

# Protocol for Windows SideShow

The Protocol for Windows SideShow, a command-based packet system, provides a standard method for transferring data between SideShow-compatible devices and PCs that are running Windows.

## Overview

Either the Windows device driver or the device can start communications. A SideShow driver can send commands to a SideShow-compatible device in the form of a packet. The device returns a packet that indicates the success or failure of the command. Additionally, the device can send event information to the driver. These events can provide notifications such as cache-content-missing notifications, application events, and user change events. The driver acknowledges receipt of the event from the device by responding with a packet.

String arguments within packets are composed of a 4‑byte value that indicates the length (in characters) of the string, followed by the string data. String data consists of 16-bit Unicode characters that are stored in big-endian byte order (bytes are numbered from left to right). Strings must not be null terminated. Buffer arguments within packets are formatted, like strings, as a 4‑byte value that indicates the length, followed by the buffer data. String or buffer arguments that contain no data are formatted as a 4‑byte value that indicates zero-length.

The bytes of a numeric value are always transmitted in little-endian order.

## Packet Header

All packets have a common header that contains the packet type, the packet number, and the size of the packet. The following structure defines the common packet header:

struct SideShowPacket

{

DWORD m\_Size; // 4 bytes

PacketType m\_Type; // 4 bytes

USHORT m\_Number // 2 bytes

## Packet Types

Each type of SideShow packet is identified by a number. The packet header must contain this number. A packet number has a 4‑byte value in which the high byte is a bit field that indicates response packets. The bit field is defined as follows:

// The packet type is a 4 byte value where the high byte is reserved // for control data.

//

// Packet type high byte bit field:

// 7 6 5 4 3 2 1 0

// --------------------------------

// | | | | | | | | |

// --------------------------------

// | | \---------------------/

// Response(1) --- | |

// | |

// NAK(1)/ACK -------- |

// |

// Error Code ----------------------

//

Bit 7 must be set if the packet is a response packet to a command. Bit 6 must be set if the response is a NAK or cleared if the response is an ACK. Bits 5 through 0 are reserved for an error code, to be included only if bit 6 is set.

## Response Packets

Response packets must be sent as a reply to command packets and event packets. A well-formed response packet header must have the following format:

* The packet number must be set to the same packet number as the command packet that is being responded to. This setting ensures that the response can be matched correctly with the command.
* The packet type must be set to the same packet type as the command packet that is being responded to. The packet type’s high bit must be set to indicate that the packet is a response packet.

The response packet type can be either ACK or NAK, as described in the following sections.

### ACK

For a positive response packet, specify ACK by clearing bit 6 of the high byte of the packet type part of the packet header. Some ACK response packets may also contain data that the command requests. The descriptions for the command packets include the specifics of which data is sent in response to a particular command.

### NAK

For a negative response packet, specify NAK by setting bit 6 of the high byte of the packet type part of the packet header. In addition, bits 5 through 0 can contain an error code for the failure of the command that is being responded to.

## Command Packets

Command packets represent the operations that the SideShow platform can request the device to perform. The driver always initiates command packets. Command packets can be composed of several pieces of data that constitute arguments for the command.

For every command, the device must return a response packet, as described in the preceding section.

The following sections describe command packet types.

### Ping

Provides a way to test a connection.

###### Packet Type ID

0x1

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK) | SideShowPacket | 10 bytes |

###### Remarks

Both devices and device drivers can use a Ping command packet to test the connection.

### SendPassThrough

Provides an extensibility mechanism that device manufacturers can use to define methods for sending custom, device-specific commands to a device.

###### Packet Type ID

0x2

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Data size | **DWORD** | 4 bytes |
| Raw data | **BYTE** | Variable |

###### Possible Values

The data buffer contains arbitrary data.

###### Response Packet Data

The response packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Data size | **DWORD** | 4 bytes |
| Raw data | **BYTE** | Variable |

### Reset

Restarts the device.

###### Packet Type ID

0x3

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| State | **DWORD** | 4 bytes |

###### Possible Values

The following table lists and describes possible values for state.

| **Value** | **Description** |
| --- | --- |
| 0 | Restart only. |
| 1 | Restart and wait in the boot-loader program. |
| Other | Manufacturer-defined values. |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

### SetUserState

Notifies user-associated devices which users are available to be selected as the owner of the device. Users must be identified by security identifiers (SIDs).

###### Packet Type ID

0x50

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Changed user SID size, in characters | **DWORD** | 4 bytes |
| Changed user SID | String (16-bit Unicode) | Variable |
| User name size, in characters | **DWORD** | 4 bytes |
| User name | String (16-bit Unicode) | Variable |
| User state | **DWORD** | 4 bytes |

###### Possible Values

The following table lists and describes possible values for user state.

| **Value** | **Description** |
| --- | --- |
| 0 | Available |
| 1 | Unavailable |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

For more information, see **GetCurrentUser** earlier in this paper.

### SetCurrentUser

Specifies the current user of the device. The user must be identified by an SID.

###### Packet Type ID

0x100

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| User SID size, in characters | **DWORD** | 4 bytes |
| User SID | String (16-bit Unicode) | Variable |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

Send this command when a user becomes the current user of the device, which typically corresponds to a computer logon. Send this command with a NULL SID when the current user is unknown, which typically corresponds to a computer logoff.

For more information, see **GetUserModel**.

### GetCurrentUser

Retrieves the SID for the current user of the device.

###### Packet Type ID

0x101

###### Command Packet Data

The command packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |

###### Response Packet Data

The response packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| User SID size, in characters | **DWORD** | 4 bytes |
| SID for the current user | String (16-bit Unicode) | Variable |

###### Remarks

The contents of the SID for the current user in the response packet varies with the user model. For Assigned user model devices, a device should return the NULL SID (S‑1-0-0) until it receives the current user’s SID from a call to **SetCurrentUser**. If the device is using the Console user model, the driver should return the well-known interactive user's SID. For more information about user models, see “User Models” in the “Windows SideShow” section of the WDK.

### GetDeviceFirmwareVersion

Retrieves a string that identifies the version of the device firmware.

###### Packet Type ID

0x102

###### Command Packet Data

The command packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |

###### Response Packet Data

The response packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Firmware version size, in characters | **DWORD** | 4 bytes |
| Firmware version | String (16-bit Unicode) | Variable |

###### Remarks

The manufacturer defines the format for the firmware version string.

### GetCapabilities

Retrieves a device capability value.

###### Packet Type ID

0x103

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Requested capability | **PROPERTYKEY** | 20 bytes |

###### Response Packet Data

The response packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Value type | **DWORD** | 4 bytes |
| Capabilities | **BYTE** | Variable, depending on the data type that is associated with the capability. See “Remarks.” |

###### Remarks

The supported **PROPVARIANT** types are VT\_EMPTY, VT\_I2, VT\_I4, VT\_R4, VT\_R8, VT\_DATE, VT\_BOOL, VT\_I1, VT\_UI1, VT\_UI2, VT\_UI4, VT\_I8, VT\_UI8, VT\_INT, VT\_UINT, VT\_LPWSTR, VT\_CLSID, and VT\_UI1 | VT\_VECTOR.

The **PROPVARIANT**s must be encoded so that the first 4 bytes is the **VARTYPE** and the remaining bytes contain the data. Standard platform capabilities are defined in the WindowsSideShow.h header, as shown in the following table.

| **Capability** | **Type** |
| --- | --- |
| SIDESHOW\_CAPABILITY\_DEVICE\_ID | VT\_LPWSTR |
| SIDESHOW\_CAPABILITY\_SCREEN\_TYPE | VT\_I4 |
| SIDESHOW\_CAPABILITY\_SCREEN\_WIDTH | VT\_UI2 |
| SIDESHOW\_CAPABILITY\_SCREEN\_HEIGHT | VT\_UI2 |
| SIDESHOW\_CAPABILITY\_COLOR\_DEPTH | VT\_UI2 |
| SIDESHOW\_CAPABILITY\_COLOR\_TYPE | VT\_I4 |
| SIDESHOW\_CAPABILITY\_DATA\_CACHE | VT\_BOOL |
| SIDESHOW\_CAPABILITY\_SUPPORTED\_LANGUAGES | VT\_LPWSTR |
| SIDESHOW\_CAPABILITY\_CURRENT\_LANGUAGE | VT\_LPWSTR |
| SIDESHOW\_CAPABILITY\_SUPPORTED\_THEMES | VT\_LPWSTR |
| SIDESHOW\_CAPABILITY\_SUPPORTED\_IMAGE\_FORMATS | VT\_LPWSTR |
| SIDESHOW\_CAPABILITY\_CLIENT\_AREA\_WIDTH | VT\_UI2 |
| SIDESHOW\_CAPABILITY\_CLIENT\_AREA\_HEIGHT | VT\_UI2 |
| SIDESHOW\_CAPABILITY\_DEVICE\_ICON | VT\_UI1 | VT\_VECTOR |

Windows SideShow-compatible devices must provide accurate responses for all standard platform capabilities. The driver can also support additional capabilities, as required. However, supported additional capabilities must always be acknowledged by using an ACK with the VT\_EMPTY type, or NAK otherwise.

For more information about device capabilities, see “Device Capabilities“ in the MSDN® Library.

### GetApplicationOrder

Retrieves a display-ordered list of SideShow gadgets that have been added to the device.

###### Packet Type ID

0x104

###### Command Packet Data

The command packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |

###### Response Packet Data

The response packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Count of applications | **DWORD** | 4 bytes |
| Application IDs | **GUID** array | Count of applications x 16 bytes |

### SetApplicationOrder

Specifies the display order for SideShow gadgets that are running on the device.

###### Packet Type ID

0x105

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Count of applications | **DWORD** | 4 bytes |
| Application IDs | **GUID** array | Count of applications x 16 bytes |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

The **GUID** array that is passed through the Application IDs parameter must be in the same order as the desired gadget display order.

### SetLanguage

Specifies the current language and font size for a device.

###### Packet Type ID

0x106

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| String buffer size, in characters | **DWORD** | 4 bytes |
| Culture code and font size | String (16-bit Unicode) | Variable |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

Culture codes are standardized for Windows. For example, the culture code for English in the United States is “en-US”. For a detailed description of culture codes, see “CultureInfo Class” in the MSDN library.

The device manufacturer must specify which languages and countries the device supports.

The language that this command specifies must be one of the set of languages that the **GetCapabilites** command returns for the SIDESHOW\_CAPABILITY\_SUPPORTED\_LANGUAGES capability.

If the command specifies an unsupported country, the device can default to the generic language (for example, from “en-US” to “en”). The device must default to English if the language is not supported.

The format of the command string is:

<*culture\_code*>:*n*

where *n* is a 1-based index value for the set of supported font sizes.

Font sizes are indexed from smallest to largest. The font size index must be less than or equal to the number of fonts that the **GetCapabilites** command returns for the SIDESHOW\_CAPABILITY\_SUPPORTED\_LANGUAGES capability.

### GetPreEnabledApplications

Retrieves a list of gadgets, specified by the device, to be enabled by default for all users of the computer.

###### Packet Type ID

0x107

###### Command Packet Data

The command packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |

###### Response Packet Data

The response packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Count of application ID and endpoint **GUID**s | **DWORD** | 4 bytes |
| Application ID and endpoints | **GUID** array | Count of application/endpoint **GUID**s x 16 bytes. Note that each application/endpoint pair consists of two **GUID**s. |

###### Remarks

The first **GUID** of each pair is an application ID. The second **GUID** of each pair is the endpoint type to which the application connects.

You must not assume that particular gadgets are installed on the PC that is running Windows.

### SetTime

Specifies the current coordinated universal time (UTC).

###### Packet Type ID

0x108

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Time | **FILETIME** | 8 bytes |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

The specified time must be UTC.

**FILETIME** is defined by the following data structure:

typedef struct \_FILETIME {  
 DWORD dwLowDateTime;  
 DWORD dwHighDateTime;

} FILETIME,   
 \*PFILETIME;

For more information, see “FILETIME Structure” in the MSDN library.

### SetShortDateFormat

Specifies the format for displaying a concise date.

###### Packet Type ID

0x109

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| String buffer size, in characters | **DWORD** | 4 bytes |
| Format | String | Variable |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

For more information, see “Day, Month, Year, and Era Format Pictures” in the MSDN library.

### SetLongDateFormat

Indicates the format for displaying a full date.

###### Packet Type ID

0x10A

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| String buffer size, in characters | **DWORD** | 4 bytes |
| Format | String (16-bit Unicode) | Variable |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

For more information, see “Day, Month, Year, and Era Format Pictures” in the MSDN library.

### SetShortTimeFormat

Specifies the format for displaying a concise time.

###### Packet Type ID

0x10B

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| String buffer size, in characters | **DWORD** | 4 bytes |
| Format | String | Variable |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

For more information, see “Hour, Minute, and Second Format Pictures” in the MSDN library.

### SetLongTimeFormat

Indicates the format for displaying a full time.

###### Packet Type ID

0x10C

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| String buffer size, in characters | **DWORD** | 4 bytes |
| Format | String (16-bit Unicode) | Variable |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

For more information, see “Hour, Minute, and Second Format Pictures” in the MSDN library.

### AddApplication

Adds a new gadget to the device.

###### Packet Type ID

0x10D

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Application ID | **GUID** | 16 bytes |
| Endpoint ID | **GUID** | 16 bytes |
| Application name size, in characters | **DWORD** | 4 bytes |
| Application name | String (16-bit Unicode) | Variable |
| Cache policy | **DWORD** | 4 bytes |
| Online only setting | **DWORD** | 4 bytes |
| Large icon buffer size, in bytes | **DWORD** | 4 bytes |
| Large icon | **BYTE** | Variable. |
| Medium icon buffer size, in bytes | **DWORD** | 4 bytes |
| Medium icon | **BYTE** | Variable |
| Small icon buffer size, in bytes | **DWORD** | 4 bytes |
| Small icon | **BYTE** | Variable |

###### Possible Values

The following table lists possible values for platform-defined Endpoint IDs.

| **Endpoint ID** | **Description** |
| --- | --- |
| {A9A5353F-2D4B-47ce-93EE-759F3A7DDA4F} | Simple Content Format (SCF) |
| {4DFF36B5-9DDE-4F76-9A2A-96435047063D} | iCalendar format |

The following table lists possible values for Cache policy.

| **Value** | **Description** |
| --- | --- |
| 0 | Keep newest. Give priority to the newest content in the cache. |
| 1 | Keep oldest. Give priority to the oldest content in the cache. |
| 2 | Keep frequently accessed. Give priority to the most frequently used content. May be combined with Keep Oldest to give priority to the oldest item when multiple items have the same access frequency. |
| 4 | Keep Recently Accessed. Give priority to the content most recently accessed (by the user). May be combined with the Keep Oldest to give priority to the oldest item when multiple items have the same last access time. |

The following table lists possible values for Online only.

| **Value** | **Description** |
| --- | --- |
| 0 | False |
| 1 | True (online only) |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

Icon images must be in a 32-bit–per-pixel (BPP) bitmap format that contain the BITMAPFILEHEADER and BITMAPINFOHEADER structures and the image bits. These images can contain per-pixel alpha values if those alpha values are available in the original image.

### DeleteApplication

Removes a previously added gadget from the device.

###### Packet Type ID

0x10E

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Application ID | **GUID** | 16 bytes |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

This command must not remove preinstalled gadgets from the device.

### DeleteAllApplications

Removes all previously added applications from the device, the content for the applications, and their notifications.

###### Packet Type ID

0x10F

###### Command Packet Data

The command packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

This command must not remove preinstalled gadgets from the device.

### AddNotification

Adds a notification that is associated with the specified application.

###### Packet Type ID

0x110

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Application ID | **GUID** | 16 bytes |
| Notification ID | **DWORD** | 4 bytes |
| Expiration time | **FILETIME** | 8 bytes |
| Notification title size, in characters | **DWORD** | 4 bytes |
| Notification title | String (16-bit Unicode) | Variable |
| Notification message size, in characters | **DWORD** | 4 bytes |
| Notification message | String (16-bit Unicode) | Variable |
| Notification image size, in bytes | **DWORD** | 4 bytes |
| Notification image | **BYTE** | Variable. The notification image must be a 32x32 or 48x48 pixel, 32-bit color (24 bits plus alpha channel) bitmap that uses the BMP file format. |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

The notification image must be in a 32-BPP bitmap format that contains the BITMAPFILEHEADER and BITMAPINFOHEADER structures and the image bits. This image can contain a per-pixel alpha value if that alpha value is available in the original image.

### DeleteNotification

Removes a notification from the specified application.

###### Packet Type ID

0x111

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Application ID | **GUID** | 16 bytes |
| Notification ID | **DWORD** | 4 bytes |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

### DeleteAllNotifications

Removes all notifications from the specified application.

###### Packet Type ID

0x112

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Application ID | **GUID** | 16 bytes |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

### SetNotificationsEnabled

Specifies whether notifications are enabled.

###### Packet Type ID

0x113

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| IsEnabled | **DWORD** | 4 bytes |

###### Possible Values

The following table lists possible values for IsEnabled.

| **Value** | **Description** |
| --- | --- |
| 0 | False (notifications disabled) |
| 1 | True (notifications enabled) |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

### AddContentItem

Adds content for the specified application/endpoint pair.

###### Packet Type ID

0x114

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Application ID | **GUID** | 16 bytes |
| Endpoint ID | **GUID** | 16 bytes |
| Content ID | **DWORD** | 4 bytes |
| Content data size, in bytes | **DWORD** | 4 bytes |
| Content data | **BYTE** | Variable |

###### Possible Values

The following table lists possible values for platform-defined Endpoint IDs.

| **Endpoint ID** | **Description** |
| --- | --- |
| {A9A5353F-2D4B-47ce-93EE-759F3A7DDA4F} | Simple Content Format (SCF) |
| {4DFF36B5-9DDE-4F76-9A2A-96435047063D} | iCalendar format |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

Each content item must be associated with a unique ID. Sending new content with a previously sent content ID results in the new content replacing the previous content.

Basic displays that present only glance-data discard all content identifiers other than content identifier 0.

### DeleteContentItem

Removes content for the specified application/endpoint pair.

###### Packet Type ID

0x115

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Application ID | **GUID** | 16 bytes |
| Endpoint ID | **GUID** | 16 bytes |
| Content ID | **DWORD** | 8 bytes |

###### Possible Values

The following table lists possible values for platform-defined Endpoint IDs.

| **Endpoint ID** | **Description** |
| --- | --- |
| {A9A5353F-2D4B-47ce-93EE-759F3A7DDA4F} | Simple Content Format (SCF) |
| {4DFF36B5-9DDE-4F76-9A2A-96435047063D} | iCalendar format |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

### DeleteAllContentItems

Removes all content items for the specified application/endpoint pair.

###### Packet Type ID

0x116

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Application ID | **GUID** | 16 bytes |
| Endpoint ID | **GUID** | 16 bytes |

###### Possible Values

The following table lists possible values for platform-defined Endpoint IDs.

| **Endpoint ID** | **Description** |
| --- | --- |
| {A9A5353F-2D4B-47ce-93EE-759F3A7DDA4F} | Simple Content Format (SCF) |
| {4DFF36B5-9DDE-4F76-9A2A-96435047063D} | iCalendar format |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

### GetSupportedEndpoints

Retrieves the list of endpoints the device supports.

###### Packet Type ID

0x117

###### Command Packet Data

The command packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |

###### Response Packet Data

The response packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Endpoint count | **DWORD** | 4 bytes |
| Endpoints | **GUID** array | Endpoint count x 16 bytes |

###### Possible Values

The following table lists possible values for platform-defined Endpoint IDs.

| **Endpoint ID** | **Description** | |
| --- | --- | --- |
| {A9A5353F-2D4B-47ce-93EE-759F3A7DDA4F} | | Simple Content Format (SCF) |
| {4DFF36B5-9DDE-4F76-9A2A-96435047063D} | | iCalendar format |

### SetTimeZone

Specifies the current time zone.

###### Packet Type ID

0x118

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Bias | **LONG** | 4 bytes |
| Standard date | **SYSTEMTIME** | 16 bytes |
| Standard bias | **LONG** | 4 bytes |
| Daylight date | **SYSTEMTIME** | 16 bytes |
| Daylight bias | **LONG** | 4 bytes |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

###### Remarks

For information about time zones, see “TIME\_ZONE\_INFORMATION Structure” in the MSDN library.

### GetDeviceName

Retrieves a string that contains the device name.

###### Packet Type ID

0x500

###### Command Packet Data

The command packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |

###### Response Packet Data

The response packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Device name size, in characters | **DWORD** | 4 bytes |
| Device name | String (16-bit Unicode) | Variable |

###### Remarks

This name corresponds to the device name that appears in Device Manager and Windows SideShow Control Panel.

### GetDeviceManufacturer

Retrieves a string that contains the name of the device manufacturer.

###### Packet Type ID

0x501

###### Command Packet Data

The command packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |

###### Response Packet Data

The response packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Device manufacturer size, in characters | **DWORD** | 4 bytes |
| Device manufacturer | String (16-bit Unicode) | Variable |

###### Remarks

This device manufacturer name corresponds to the manufacturer name that appears in Device Manager.

### Sync

A **Sync** packet is the first packet that a device receives each time that the transport connection is opened. The **Sync** packet notifies a device that a protocol session has begun. When a device receives this command packet, the device must verify that the **GUID** corresponds to the protocol version that the device supports. If the device supports the protocol, the device must clear any data that is stored in the receive and send buffers and then send a response packet that contains the same **GUID**. The driver then resets the packet numbers to 1. If the device does not support the protocol, the device must reply with a NAK packet.

**Sync** packets always have a packet number of 0.

###### Packet Type ID

0x502

###### Command Packet Data

The command packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Synchronization value | **GUID** | 16 bytes |

###### Possible Values

The following table shows the synchronization value.

| **Value** | **Description** |
| --- | --- |
| {a33f248b-882f-4531-82c2-ed3b90c5c520} | Standard protocol |

###### Response Packet Data

The response packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Synchronization value (same as command data) | **GUID** | 16 bytes |

## Event Packets

Event packets represent events that the device needs the SideShow platform to be aware of. Devices send event packets to the driver.

The following sections describe event packet types.

### ContentMissing

Provides notification about missing content for the specified application/endpoint pair.

###### Packet Type ID

0x4000

###### Event Packet Data

The event packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Application ID | **GUID** | 16 bytes |
| Endpoint ID | **GUID** | 16 bytes |
| Content ID | **DWORD** | 4 bytes |

###### Possible Values

The following table lists possible values for platform-defined Endpoint IDs.

| **Endpoint ID** | **Description** |
| --- | --- |
| {A9A5353F-2D4B-47ce-93EE-759F3A7DDA4F} | Simple Content Format (SCF) |
| {4DFF36B5-9DDE-4F76-9A2A-96435047063D} | iCalendar format |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

### ApplicationEvent

Provides an application-defined event notification.

###### Packet Type ID

0x4001

###### Event Packet Data

The event packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| Application ID | **GUID** | 16 bytes |
| Endpoint ID | **GUID** | 16 bytes |
| Event ID | **DWORD** | 4 bytes |
| Event data size, in bytes | **DWORD** | 4 bytes |
| Event data | **BYTE** | Variable |

###### Possible Values

The following table lists possible values for platform-defined Endpoint IDs.

| **Endpoint ID** | **Description** |
| --- | --- |
| {A9A5353F-2D4B-47ce-93EE-759F3A7DDA4F} | Simple Content Format (SCF) |
| {4DFF36B5-9DDE-4F76-9A2A-96435047063D} | iCalendar format |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

### ChangeUserRequestEvent

Requests that a new, available user take ownership of the device. The new user must be identified by a SID.

###### Packet Type ID

0x4002

###### Event Packet Data

The event packet must contain the data in the following table, in the order shown.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header | SideShowPacket | 10 bytes |
| User SID size, in characters | **DWORD** | 4 bytes |
| User SID | String (16-bit Unicode) | Variable |

###### Response Packet Data

The response packet must contain the data in the following table.

| **Data** | **Type** | **Size** |
| --- | --- | --- |
| Standard header (ACK or NAK) | SideShowPacket | 10 bytes |

# Resources

#### Windows Driver Kit:

Windows SideShow

<http://msdn2.microsoft.com/en-us/library/bb870497.aspx>

User Models

<http://msdn.microsoft.com/en-us/library/aa939009.aspx>

#### MSDN:

Day, Month, Year, and Era Format Pictures

<http://msdn.microsoft.com/en-us/library/ms776306(vs.85).aspx>

Device Capabilities

<http://msdn.microsoft.com/en-us/library/ms744040(VS.85).aspx>

CultureInfo Class

<http://msdn.microsoft.com/en-us/library/system.globalization.cultureinfo.aspx>

FILETIME Structure

<http://msdn.microsoft.com/en-us/library/ms724284.aspx>

Hour, Minute, and Second Format Pictures

<http://msdn.microsoft.com/en-us/library/ms776320(VS.85).aspx>

TIME\_ZONE\_INFORMATION Structure

<http://msdn.microsoft.com/en-us/library/ms725481.aspx>

#### WHDC Web site:

Microsoft OS Descriptors

<https://www.microsoft.com/whdc/connect/usb/os_desc.mspx>

PNP-X: Plug and Play Extensions for Windows Specification

<http://www.microsoft.com/whdc/connect/rally/pnpx-spec.mspx>