Digitizer Drivers for Windows Touch and Pen-Based Computers

October 21, 2010

Abstract

This paper provides information about digitizer drivers for Windows® Touch and pen-based computers for the Windows family of operating systems. Windows Touch is the name for the touch and multitouch capabilities that are built into the Windows platform. The paper also describes how Window interfaces with digitizer drivers for human interface devices (HID) and includes guidelines for driver writers.

This information applies to the following operating systems:   
 Windows 7

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/input/DigitizerDrvs_touch.mspx>

|  |
| --- |
| **Disclaimer**: This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.  This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.  © 2010 Microsoft Corporation. All rights reserved. |

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| October 21, 2010 | Revised to highlight Windows 7 Touch technology. | | | |
| October 19, 2009 | Revised to clarify Touch requirements for Windows Vista. | | | |
| March 16, 2009 | Revised after multitouch HID specification proposal ratified. | | | |
| December 18, 2008 | Previously titled ”Digitizer Drivers for Touch, Multitouch, and Pen Devices”. | | | |
| November 5, 2008 | First publication | | | |

Contents

[Introduction 3](#_Toc275273928)

[Touch Digitizer Drivers 3](#_Toc275273929)

[Required and Optional HID Usages 3](#_Toc275273930)

[Sample Report Descriptor for a Touch Digitizer Device 4](#_Toc275273931)

[Multitouch Digitizer Drivers 5](#_Toc275273932)

[Required and Optional HID Usages 5](#_Toc275273933)

[HID Specification Support for Multitouch 6](#_Toc275273934)

[Reporting Modes 7](#_Toc275273935)

[Serial Mode 7](#_Toc275273936)

[Parallel Mode 7](#_Toc275273937)

[Hybrid Mode 8](#_Toc275273938)

[Null Values 8](#_Toc275273939)

[Sample Report Descriptor (Parallel/Hybrid Mode) 8](#_Toc275273940)

[Sample Report Descriptor (Serial Reporting Mode) 10](#_Toc275273941)

[Pen Digitizer Drivers 11](#_Toc275273942)

[Required and Optional HID Usages 11](#_Toc275273943)

[Sample Report Descriptor for a Pen Digitizer Device 11](#_Toc275273944)

[Dual-mode Digitizers 12](#_Toc275273945)

[Reporting Physical Dimensions 12](#_Toc275273946)

[Top-Level Collection Guidance 12](#_Toc275273947)

[Capability Discovery and Configuration 13](#_Toc275273948)

[Device 13](#_Toc275273949)

[Feature Report Requirements 13](#_Toc275273950)

[Device Mode 14](#_Toc275273951)

[Device Identifier 14](#_Toc275273952)

[Touch and Pen Support 15](#_Toc275273953)

[Resources 15](#_Toc275273954)

# Introduction

Windows® Touch is the name for the touch and multitouch capabilities that are built into the Windows family of operating systems. In the context of Windows Touch, *touch* refers to support of a single physical contact point, whereas *multitouch* refers to support for two or more concurrent physical contacts. Support for touch capabilities was introduced in Windows Vista, and support for multitouch capabilities was introduced in Windows 7.

Touch and multitouch are becoming more prevalent on devices. From Microsoft Surface to home-brewed multitouch devices, it is clear that hardware availability is beginning to catch up to user demand for natural interfaces and intuitive computing experiences.

This white paper covers how Windows Vista and Windows 7 interface with human interface device (HID) digitizer devices. This paper assumes familiarity with the *Device Class Definition for Human Interface Devices (HID)* specification and the *HID Usage Tables* that are published by the USB Implementers Forum. For pointers to HID information, see “Resources” at the end of this paper.

# Touch Digitizer Drivers

Devices that appear through HID as touch digitizers are intended to be used with the finger, not a stylus. The operating system provides optimized behavior for different input methods. For example, touch devices receive gestures that are tuned for the finger.

## Required and Optional HID Usages

Starting with Windows Vista, touch digitizers appear through HID as a touch digitizer (page 0x0D, usage 0x04).

The following usages are required:

* X (page 0x01, usage 0x30) and Y (page 0x01, usage 0x31).
* Tip switch (page 0x0D, usage 0x42).
* In-range (page 0x0D, usage 0x32).

Use tip switch to indicate finger contact and liftoff from the digitizer surface, similar to how a pen reports contact with the digitizer.

If the device supports z-axis detection, it reports in-range correctly. If the device does not support z-axis detection, the driver reports packets with in-range and tip switch set when a finger comes in contact with the digitizer.

Note that Windows XP has different guidelines for how touch digitizer drivers should handle in-range reporting. The Windows XP Tablet PC OEM Preinstallation Kit (OPK) includes information about how to implement a touch digitizer driver for Windows XP. For more information, contact your Microsoft technical account manager.

The following usages are optional, but you should implement them if the digitizer hardware supports them:

* Confidence (page 0x0D, usage 0x47).
* Width and height (page 0x0D, usages 0x48 and 0x49).

Confidence is a suggestion from the device regarding whether the touch contact was an intended or accidental touch. Your device should reject accidental touches as thoroughly as it can and report that information by using the confidence usage.

Starting with Windows Vista, the operating system uses confidence to help improve accidental touch rejection. In addition to the confidence value, the operating system applies additional heuristics to the touch input stream to improve accidental touch rejection.

If your device does not report confidence, then it is entirely up to your device to provide accidental touch rejection.

The width and height usages represent the width and height of the touch contact. Width and height are also exposed to application developers through the Windows Touch platform.

## Sample Report Descriptor for a Touch Digitizer Device

Here is a sample report descriptor for a touch digitizer device:

0x05, 0x0d,                       // USAGE\_PAGE (Digitizers)          0

    0x09, 0x04,                         // USAGE (Touch Screen)             2

    0xa1, 0x01,                         // COLLECTION (Application)         4

    0x85, REPORTID\_TOUCH,               //   REPORT\_ID (Touch)              6

    0x09, 0x20,                         //   USAGE (Stylus)                 8

    0xa1, 0x00,                         //   COLLECTION (Physical)          10

    0x09, 0x42,                         //     USAGE (Tip Switch)           12

    0x15, 0x00,                         //     LOGICAL\_MINIMUM (0)          14

    0x25, 0x01,                         //     LOGICAL\_MAXIMUM (1)          16

    0x75, 0x01,                         //     REPORT\_SIZE (1)              18

    0x95, 0x01,                         //     REPORT\_COUNT (1)             20

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         22

    0x95, 0x03,                         //     REPORT\_COUNT (3)             24

    0x81, 0x03,                         //     INPUT (Cnst,Ary,Abs)         26

    0x09, 0x32,                         //     USAGE (In Range)             28

    0x09, 0x37,                         //     USAGE (Data Valid-Finger)    30

    0x95, 0x02,                         //     REPORT\_COUNT (2)             32

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         34

    0x95, 0x0a,                         //     REPORT\_COUNT (10)            36

    0x81, 0x03,                         //     INPUT (Cnst,Ary,Abs)         38

    0x05, 0x01,                         //     USAGE\_PAGE (Generic Desktop) 40

    0x26, 0xff, 0x7f,                   //     LOGICAL\_MAXIMUM (32767)      42

    0x75, 0x10,                         //     REPORT\_SIZE (16)             45

    0x95, 0x01,                         //     REPORT\_COUNT (1)             47

    0xa4,                               //     PUSH                         49

    0x55, 0x0d,                         //     UNIT\_EXPONENT (-3)           50

    0x65, 0x00,                         //     UNIT (None)                  52

    0x09, 0x30,                         //     USAGE (X)                    54

    0x35, 0x00,                         //     PHYSICAL\_MINIMUM (0)         56

    0x46, 0x00, 0x00,                   //     PHYSICAL\_MAXIMUM (0)         58

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         61

    0x09, 0x31,                         //     USAGE (Y)                    63

    0x46, 0x00, 0x00,                   //     PHYSICAL\_MAXIMUM (0)         65

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         68

    0xb4,                               //     POP                          70

    0x05, 0x0d,                         //     USAGE PAGE (Digitizers)      71

    0x09, 0x60,                         //     USAGE (Width)                73

    0x09, 0x61,                         //     USAGE (Height)               75

    0x95, 0x02,                         //     REPORT\_COUNT (2)             77

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         79

    0x95, 0x01,                         //     REPORT\_COUNT (1)             81

    0x81, 0x03,                         //     INPUT (Cnst,Ary,Abs)         83/85

    0xc0,                               //   END\_COLLECTION                 0/1

    0xc0,                               // END\_COLLECTION                   0/1

# Multitouch Digitizer Drivers

## Required and Optional HID Usages

Starting with Windows 7, multitouch digitizers appear as HID touch digitizers (page 0x0D, usage 0x04), but they also contain the contact ID usage in their report descriptor (page 0x0D, usage 0x51).

Multitouch digitizer drivers must implement the following usages:

* X (page 0x01, usage 0x30) and Y (page 0x01, usage 0x31).
* Contact ID (page 0x0D, usage 0x51).
* Contact count maximum (page 0x0D, usage 0x55).
* Tip switch (page 0x0D, usage 0x42).
* In-range (page 0x0D, usage 0x32).

The following usages are optional, but you should implement them if the digitizer hardware supports them:

* Confidence (page 0x0D, usage 0x47).
* Width and height (page 0x0D, usages 0x48 and 0x49).

## HID Specification Support for Multitouch

The HID specification now supports multitouch input from digitizers, and we implement these features in Windows 7.

Proposed Additions to the HID Specification to Support Multitouch

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Description | Page | Type | ID |
| Contact identifier | Contact identifier | Digitizer | Dynamic Value (DV) | 0x51 |
| Configuration | Configuration | Digitizer | Collection Application (CA) | 0x0E |
| Device mode | Input mode | Digitizer | DV | 0x52 |
| Device settings | Device setting | Digitizer | Collection Logical (CL) | 0x23 |
| Device identifier | Device index | Digitizer | DV | 0x53 |
| Contact count | Actual contact count | Digitizer | DV | 0x54 |
| Contact count maximum | Maximum number of contacts supported | Digitizer | DV | 0x55 |

Contact identifier

The ID of the current contact. An ID must remain constant while it is detected by the device, and each separate concurrent contact must have a unique ID. IDs can be reused after a contact is no longer detected. If the device supports “in-air” packets, the ID must persist from the time that the contact is detected until the time that it goes out of range.

Configuration

The Collection Application for the top-level collection that houses the configuration feature report.

Device mode

A read/write value feature to get and set the current input configuration of a device.

Device settings

The logical collection that contains the device configuration usages (Device identifier and Device mode).

Device identifier

This is a static value (SV) when it is part of a digitizer or mouse top-level collection. It is required if multiple digitizer top-level collections of the same kind in a report descriptor exist. This usage uniquely identifies the digitizer top-level collection and should appear in a feature report. If the device can function as a mouse, the mouse collection should have the same device identifier as the corresponding digitizer collection. Devices with only one digitizer top-level collection do not require a device identifier usage.

Contact count

The usage that is specifically for devices that report all current contact information by using the parallel or hybrid mode that is described later in this paper. Because the report count for each usage in this device’s top-level collection is less than or equal to the maximum number of contacts that the device supports, this usage should convey the actual number of valid contacts in the current packet. A device that cannot provide this information must use NULL for all values in the first position that do not contain valid contact information. The usage should be in the top-level collection.

Contact count maximum

A read-only feature for getting the total number of contacts that a multitouch device supports. This usage must be included in a feature report that is part of the multitouch top-level collection regardless of the reporting mode that is being used. (See the sample descriptors later in this paper.)

## Reporting Modes

Windows supports three ways of reporting multitouch data to the system:

* **Serial mode:** Each packet contains a single contact update.
* **Parallel mode:** Each packet is “wide” enough to support reporting the maximum number of contacts that the device can detect.
* **Hybrid mode:** Each packet contains a fixed number of contacts (less than the total number that the device can detect), and multiple packets are sent to convey all the contact data.

### Serial Mode

In this mode, devices report all contact information in a series of packets. Each packet contains a single contact update, and the device sends a separate packet for each concurrent contact. For example, if two fingers are down, a device that uses serial mode sends an update for the first contact, then sends an update for the second contact, and then repeats while both fingers are down.

One of the challenges of serial mode is that it can affect the device’s per-contact reporting speeds. For example, if a device can send an update once every millisecond (ms) and if two contacts are down, then any given contact is reported only every 2 ms.

Hybrid and parallel reporting modes have the advantage of reducing the data delivery overhead. We recommend that you consider one of these alternative modes if it is practical because these modes can improve the efficiency of data delivery from operating system component to applications.

### Parallel Mode

In this mode, devices report all contact information in one packet. Each contact is represented by a logical collection that is embedded in the top-level collection. This logical collection contains all the usages that the device supports for each contact (for example, X, Y, and pressure). Because the device generally reports fewer contacts than the maximum, the number of contacts that are reported in a parallel packet should be conveyed either in the actual contacts usage (see later in this paper) or by setting null values for all in‑valid contacts in a packet.

Consider a device that supports three contacts. If the user currently has only two fingers on the digitizer, the parallel packet has only two valid contact data in the packet that can carry data, for a maximum of three contacts. In this case, the actual count should be set to two to let the client application know that the information beyond the second contact data is not valid. Alternatively, the device can set the values of the contact usages beyond the second entry to null values. The client application can detect the actual contacts either by examining the value of the actual count usage or reading the data until it hits a null value.

Reporting multiple contacts in one packet has the disadvantage of wasting space per packet every time there are fewer contacts to report than the maximum. Devices can use a hybrid mode for reporting data to reduce this inefficiency.

### Hybrid Mode

In hybrid mode, the numbers of contacts that can be reported in one packet is less than the maximum that the device supports. For example, a device that supports a maximum of 48 contacts can set up its top-level collection to report a maximum of 12 contacts in one packet. If it must report 48 contacts, it can break these down into 4 serial packets that report 12 contacts each.

When a device chooses to report data in this manner, the actual contact usage value in the first packet should reflect the total number of contacts that are being reported in the hybrid packets. The other serial packets should have an actual count of 0. Using the preceding example, the actual count usage in the first packet has a value of 48, whereas the subsequent three packets have an actual usage count of 0.

### Null Values

Null values should be specified as outlined in the HID Specification. The null bit must be set on all main items in the report descriptor. Note that a device can use either the actual count usage or null values to notify the host of the actual number of valid contacts in a packet.

## Sample Report Descriptor (Parallel/Hybrid Mode)

This sample report descriptor could easily be turned into a parallel or hybrid report, depending on the relationship between the maximum count and the actual count:

0x05, 0x0d, // USAGE\_PAGE (Digitizers)

0x09, 0x04, // USAGE (Touch Screen)

0xa1, 0x01, // COLLECTION (Application)

0x85, REPORTID\_MTOUCH, // REPORT\_ID (Touch)

0x09, 0x22, // USAGE (Finger)

0xa1, 0x02, // COLLECTION (Logical)

0x09, 0x42, // USAGE (Tip Switch)

0x15, 0x00, // LOGICAL\_MINIMUM (0)

0x25, 0x01, // LOGICAL\_MAXIMUM (1)

0x75, 0x01, // REPORT\_SIZE (1)

0x95, 0x01, // REPORT\_COUNT (1)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x09, 0x32, // USAGE (In Range)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x09, 0x47, // USAGE (Touch Valid)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x95, 0x05, // REPORT\_COUNT (5)

0x81, 0x03, // INPUT (Cnst,Ary,Abs)

0x75, 0x08, // REPORT\_SIZE (8)

0x09, 0x51, // USAGE (Contact Identifier)

0x95, 0x01, // REPORT\_COUNT (1)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x05, 0x01, // USAGE\_PAGE (Generic Desk..

0x26, 0xff, 0x7f, // LOGICAL\_MAXIMUM (32767)

0x75, 0x10, // REPORT\_SIZE (16)

0x55, 0x00, // UNIT\_EXPONENT (0)

0x65, 0x00, // UNIT (None)

0x09, 0x30, // USAGE (X)

0x35, 0x00, // PHYSICAL\_MINIMUM (0)

0x46, 0x00, 0x00, // PHYSICAL\_MAXIMUM (0)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x09, 0x31, // USAGE (Y)

0x46, 0x00, 0x00, // PHYSICAL\_MAXIMUM (0)

0x81, 0x02, // INPUT (Data,Var,Abs)

0xc0, // END\_COLLECTION

0xa1, 0x02, // COLLECTION (Logical)

0x05, 0x0d, // USAGE\_PAGE (Digitizers)

0x09, 0x42, // USAGE (Tip Switch)

0x15, 0x00, // LOGICAL\_MINIMUM (0)

0x25, 0x01, // LOGICAL\_MAXIMUM (1)

0x75, 0x01, // REPORT\_SIZE (1)

0x95, 0x01, // REPORT\_COUNT (1)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x09, 0x32, // USAGE (In Range)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x09, 0x47, // USAGE (Touch Valid)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x95, 0x05, // REPORT\_COUNT (5)

0x81, 0x03, // INPUT (Cnst,Ary,Abs)

0x75, 0x08, // REPORT\_SIZE (8)

0x09, 0x51, // USAGE ( Cotact Identifier)

0x95, 0x01, // REPORT\_COUNT (1)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x05, 0x01, // USAGE\_PAGE (Generic Desk..

0x26, 0xff, 0x7f, // LOGICAL\_MAXIMUM (32767)

0x75, 0x10, // REPORT\_SIZE (16)

0x55, 0x00, // UNIT\_EXPONENT (0)

0x65, 0x00, // UNIT (None)

0x09, 0x30, // USAGE (X)

0x35, 0x00, // PHYSICAL\_MINIMUM (0)

0x46, 0x00, 0x00, // PHYSICAL\_MAXIMUM (0)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x09, 0x31, // USAGE (Y)

0x46, 0x00, 0x00, // PHYSICAL\_MAXIMUM (0)

0x81, 0x02, // INPUT (Data,Var,Abs)

0xc0, // END\_COLLECTION

0x05, 0x0d, // USAGE\_PAGE (Digitizers)

0x09, 0x54, // USAGE (Contact Count)

0x95, 0x01, // REPORT\_COUNT (1)

0x75, 0x08, // REPORT\_SIZE (8)

0x15, 0x00, // LOGICAL\_MINIMUM (0)

0x25, 0x08, // LOGICAL\_MAXIMUM (8)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x09, 0x55, // USAGE(Contact Count Maximum)

0xb1, 0x02, // FEATURE (Data,Var,Abs)

0xc0, // END\_COLLECTION

The report descriptor has a top-level collection with two embedded logical collections. Each one represents data that can be received from each contact detected. Note that the actual count usage is not in either logical collection. This report descriptor lets the device report all contact information (in this case, a maximum of two) in one packet.

The presence of an actual count that is greater than the number of contacts that fit into one packet indicates to the client application that the device is using the hybrid data reporting format.

## Sample Report Descriptor (Serial Reporting Mode)

The following is a sample report descriptor for serial reporting mode:

0x05, 0x0d, // USAGE\_PAGE (Digitizers)

0x09, 0x04, // USAGE (Touch Screen)

0xa1, 0x01, // COLLECTION (Application)

0x85, REPORTID\_MTOUCH, // REPORT\_ID (Touch)

0x09, 0x22, // USAGE (Finger)

0xa1, 0x00, // COLLECTION (Physical)

0x09, 0x42, // USAGE (Tip Switch)

0x15, 0x00, // LOGICAL\_MINIMUM (0)

0x25, 0x01, // LOGICAL\_MAXIMUM (1)

0x75, 0x01, // REPORT\_SIZE (1)

0x95, 0x01, // REPORT\_COUNT (1)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x95, 0x03, // REPORT\_COUNT (3)

0x81, 0x03, // INPUT (Cnst,Ary,Abs)

0x09, 0x32, // USAGE (In Range)

0x09, 0x47, // USAGE (Touch Valid)

0x95, 0x02, // REPORT\_COUNT (2)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x95, 0x0a, // REPORT\_COUNT (10)

0x81, 0x03, // INPUT (Cnst,Ary,Abs)

0x05, 0x01, // USAGE\_PAGE (Generic Desk..

0x26, 0xff, 0x7f, // LOGICAL\_MAXIMUM (32767)

0x75, 0x10, // REPORT\_SIZE (16)

0x95, 0x01, // REPORT\_COUNT (1)

0x65, 0x00, // UNIT (None)

0x09, 0x30, // USAGE (X)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x09, 0x31, // USAGE (Y)

0x46, 0x00, 0x00, // PHYSICAL\_MAXIMUM (0)

0x81, 0x02, // INPUT (Data,Var,Abs)

0xb4, // POP

0x05, 0x0d, // USAGE PAGE (Digitizers)

0x09, 0x48, // USAGE (Width)

0x09, 0x49, // USAGE (Height)

0x95, 0x01, // REPORT\_COUNT (2)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x81, 0x03, // INPUT (Cnst,Ary,Abs)

0x09, 0x51, // USAGE (Contact Identifier)

0x75, 0x10, // REPORT\_SIZE (16)

0x95, 0x02, // REPORT\_COUNT (1)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x09, 0x55, // USAGE(Contact Count Maximum)

0x15, 0x00, // LOGICAL\_MINIMUM (0)

0x25, 0x08, // LOGICAL\_MAXIMUM (8)

0x75, 0x08, // REPORT\_SIZE (8)

0x95, 0x01, // REPORT\_COUNT (1)

0xb1, 0x02, // FEATURE (Data,Var,Abs)

0xc0, // END\_COLLECTION

0xc0, // END\_COLLECTION

# Pen Digitizer Drivers

Starting with Windows Vista, devices that appear as HID pen devices are typically electromagnetic (EM) pen devices or devices that are intended to be used with a stylus. The operating system provides optimized behavior for different input methods. For example, pen devices receive gestures that are tuned for a stylus.

## Required and Optional HID Usages

Pen digitizer drivers must implement the following HID usages:

* X (page 0x01, usage 0x30) and Y (page 0x01, usage 0x31).
* Tip switch (page 0x0D, usage 0x42).
* In-range (page 0x0D, usage 0x32).

The following usages are optional, but we recommend that you implement them if the digitizer hardware supports them:

* Pressure (page 0x0D, usage 0x30).
* Tilt (page 0x0D, usages 0x3D and 0x3E).
* Inverted (page 0x0D, usage 0x3C).
* Barrel button (page 0x0D, usage 0x44).

You may need to add a second top-level collection for a mouse if the driver is used on slate Tablet PCs or other computers without a physical mouse. Starting with Windows Vista, if the operating system does not detect a mouse, it does not show a cursor. By adding a second collection for a mouse, you force Windows to show the cursor. The mouse collection should not send input.

For a sample pen digitizer driver, see the Wacom digitizer driver sample (Wacomopen.sys) in the Windows Driver Kit (WDK). Within the WDK installation directory, this sample driver is located at src\input\hiddigi.

## Sample Report Descriptor for a Pen Digitizer Device

The following is a sample report descriptor for a pen digitizer device:

0x05, 0x0d,                         // USAGE\_PAGE (Digitizers)          0

    0x09, 0x02,                         // USAGE (Pen)                      2

    0xa1, 0x01,                         // COLLECTION (Application)         4

    0x85, REPORTID\_PEN,                 //   REPORT\_ID (Pen)                6

    0x09, 0x20,                         //   USAGE (Stylus)                 8

    0xa1, 0x00,                         //   COLLECTION (Physical)          10

    0x09, 0x42,                         //     USAGE (Tip Switch)           12

    0x09, 0x44,                         //     USAGE (Barrel Switch)        14

    0x09, 0x45,                         //     USAGE (Eraser Switch)        16

    0x09, 0x3c,                         //     USAGE (Invert)               18

    0x09, 0x32,                         //     USAGE (In Range)             20

    0x15, 0x00,                         //     LOGICAL\_MINIMUM (0)          22

    0x25, 0x01,                         //     LOGICAL\_MAXIMUM (1)          24

    0x75, 0x01,                         //     REPORT\_SIZE (1)              26

    0x95, 0x05,                         //     REPORT\_COUNT (5)             28

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         30

    0x95, 0x0b,                         //     REPORT\_COUNT (11)            32

    0x81, 0x03,                         //     INPUT (Cnst,Var,Abs)         34

    0x05, 0x01,                         //     USAGE\_PAGE (Generic Desktop) 36

    0x26, 0xff, 0x7f,                   //     LOGICAL\_MAXIMUM (32767)      38

    0x75, 0x10,                         //     REPORT\_SIZE (16)             41

    0x95, 0x01,                         //     REPORT\_COUNT (1)             43

    0xa4,                               //     PUSH                         45

    0x55, 0x0d,                         //     UNIT\_EXPONENT (-3)           46

    0x65, 0x33,                         //     UNIT (Inch,EngLinear)        48

    0x09, 0x30,                         //     USAGE (X)                    50

    0x35, 0x00,                         //     PHYSICAL\_MINIMUM (0)         52

    0x46, 0x00, 0x00,                   //     PHYSICAL\_MAXIMUM (0)         54

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         57

    0x09, 0x31,                         //     USAGE (Y)                    59

    0x46, 0x00, 0x00,                   //     PHYSICAL\_MAXIMUM (0)         61

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         64

    0xb4,                               //     POP                          66

    0x05, 0x0d,                         //     USAGE\_PAGE (Digitizers)      67

    0x09, 0x30,                         //     USAGE (Tip Pressure)         69

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         71

    0x09, 0x3d,                         //     USAGE (X Tilt)               73

    0x09, 0x3e,                         //     USAGE (Y Tilt)               75

    0x16, 0x01, 0x80,                   //     LOGICAL\_MINIMUM (-32767)     77

    0x95, 0x02,                         //     REPORT\_COUNT (2)             80

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         82/84

#ifdef \_CUSTOMVENDORUSAGE\_

    0x06, 0x00, 0xff,                   //     USAGE\_PAGE (Vendor Defined)  0

    0x09, 0x80,                         //     USAGE (Vendor Usage 0x80)    3

    0x09, 0x81,                         //     USAGE (Vendor Usage 0x81)    5

    0x15, 0x00,                         //     LOGICAL\_MINIMUM (0)          7

    0x27, 0xff, 0xff, 0xff, 0xff,       //     LOGICAL\_MAXIMUM (0xffffffff) 9

    0x75, 0x20,                         //     REPORT\_SIZE (32)             14

    0x95, 0x02,                         //     REPORT\_COUNT (2)             16

    0x81, 0x02,                         //     INPUT (Data,Var,Abs)         18/20

#endif

    0xc0,                               //   END\_COLLECTION                 0/1

    0xc0,                               // END\_COLLECTION                   0/1

# Dual-mode Digitizers

If your device includes a digitizer that provides both pen and touch functionality, your driver must report pen and touch collections separately according to the preceding pen and touch guidance. If your driver might be installed on a slate Tablet PC or other computer without a mouse, your driver should also report a mouse collection.

# Reporting Physical Dimensions

A digitizer device reports correct physical dimensions by using the physical dimensions global items of the x and y usage. Starting with Windows Vista, the operating system uses these values in gesture recognition and other platform features. Inaccurate information about the physical dimensions may affect the performance of those features.

If a digitizer device cannot report accurate physical dimensions, it reports 0 units. (See the preceding report descriptor.)

# Top-Level Collection Guidance

All multiple input devices must include top-level collections in their report descriptor for the primary device and for a mouse. For devices that must work with other operating systems, the report descriptor must also include a top-level collection that contains a feature report that can be used to configure the device to act as a multi-input, single-input, or mouse device. (For details, see “[Capability Discovery and Configuration](#_Capability_Discovery_and)” later in this paper.)

All input reports in the digitizer usage page with Collection Applications (CAs) 1, 2, 4, or 5 (Digitizer, Pen, Touch Screen, and Touch Pad, respectively) can support multiple inputs. Windows only supports CAs 1, 2, and 4.

Devices can send multitouch data by using one report for each contact, or they can take advantage of the reporting modes that were outlined earlier in this paper for a more efficient way to report data. Report descriptors for touch devices should use Finger (0x22) CL (Collection Logical) to group the data and control usages in the top-level collections, whereas the Stylus (0x20) CL should be used to group pen-related control and data usages.

# Capability Discovery and Configuration

This section describes capability discovery and provides guidance for reporting device capabilities.

## Device

The operating system performs capability discovery by using the report descriptor. A device must report its device capabilities to the operating system by using a report descriptor after the driver is loaded.

For Windows 7 to detect a device’s ability to support multi-input, the driver must include the contact identifier usage in the report descriptor. The operating system opens the configuration top-level collection that contains the device mode feature report for exclusive access. The operating system can reconfigure the device to report data by using the multi-input, single-input, or mouse top-level collection. Because the operating system opens the feature report exclusively, the report is not accessible to third-party applications in Windows 7.

**Note:**In Windows XP and Windows Vista®, third-party applications can use the feature report to configure the device to use single touch or the mouse as they want. We recommend single touch for Windows XP Tablet PC Edition and Windows Vista. We recommend the mouse for Windows XP and Windows 2000.

## Feature Report Requirements

The feature report must be in its own top-level collection and must include the multi-input configuration usages.

The following is a typical feature report:

0x09, 0x0E,                         // USAGE (Device Configuration)

    0xa1, 0x01,                         // COLLECTION (Application)

    0x85, REPORTID\_FEATURE,             //   REPORT\_ID (Configuration)

    0x09, 0x23,                         //   USAGE (Device Settings)

    0xa1, 0x02,                         //   COLLECTION (logical)

    0x09, 0x52,                         //    USAGE (Device Mode)

    0x09, 0x53,                         //    USAGE (Device Identifier)

    0x15, 0x00,                         //    LOGICAL\_MINIMUM (0)

    0x25, 0x0a,                         //    LOGICAL\_MAXIMUM (10)

    0x75, 0x08,                         //    REPORT\_SIZE (8)

    0x95, 0x02,                         //    REPORT\_COUNT (2)

    0xb1, 0x02,                         //   FEATURE (Data,Var,Abs

    0xc0,                               //   END\_COLLECTION

    0xc0,                               // END\_COLLECTION

## Device Mode

The device mode usage can have one of the following values:

|  |  |
| --- | --- |
| Mode | Value |
| Mouse (recommended default) | 0x00 |
| Single-input (single touch or stylus) | 0x01 |
| Multi-input | 0x02 |

Using configuration modes ensures that multi-input devices can be used as single-input devices on operating systems that do not support multi-input devices.

When single-touch mode is set, device developers may do one of the following:

* Send information about the first contact only. This can be implemented in the firmware or the device driver.
* Drop all information about other contacts in the HID minidriver. This approach reduces the logic that is required in the firmware. However, it is available only to implementers who choose to write a device driver.

When mouse mode is set, the firmware or device driver should route the data from the first contact that was detected through the mouse top-level collection.

Choose the appropriate default device mode for your device based on your device’s capabilities and the operating system versions that it supports. For broad backward compatibility, we recommend mouse mode as the default. With the default set to mouse mode, the device can work with any operating system.

**Note:**If you can guarantee that your device will not be used on any operating system earlier than Windows Vista, then single-input is a better choice. Starting with Windows 7, the operating system reconfigures the device for multi-input if it discovers the capability.

## Device Identifier

This is a static value (SV) when it is part of a digitizer or mouse top-level collection. It is required when a report descriptor contains multiple digitizer top-level collections of the same kind. This usage uniquely identifies the digitizer top-level collection and should appear in a feature report. If the device can function as a mouse, the mouse collection should have the same device identifier as the corresponding digitizer collection. Devices with only one digitizer top-level collection are not required to have a device identifier usage.

When the usage is part of a device settings logical collection, it is a dynamic value (DV). It allows the host to select the device it wants to configure. A value of 0 indicates all collections. A nonzero value indicates the top-level collection with matching device identifier.

# Touch and Pen Support

If your device includes a digitizer that provides both Windows Touch and pen functionality, you must report touch and pen collections separately according to the guidance in this paper. If your driver might be installed on a slate Tablet PC or other computer that has no mouse, you should also report a mouse collection.

# Resources

If you have additional questions that are not addressed by this document, send e‑mail to [tab-ext@microsoft.com](mailto:tab-ext@microsoft.com). The following links provide further information on input devices and related topics.

The USB Implementers Forum

http://www.usb.org

Device Class Definitions for Human Interface Devices (HID)

<http://www.usb.org/developers/devclass_docs/HID1_11.pdf>

HID Usage Tables

<http://www.usb.org/developers/devclass_docs/Hut1_12.pdf>

Windows Driver Kit (WDK)

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

Windows Logo Program for Hardware

<http://www.microsoft.com/whdc/winlogo/hwrequirements.mspx>

WDK Wacom Digitizer Driver Sample (Wacomopen.sys)

This sample is located within the WDK installation directory at src\input\hiddigi.