USB Feature Specification: Notifications

INTEL^{Ò} CORPORATION

Revision 1.0

October 27, 1999

Revision History

Revision	Issue Date	Comments
1.0	10/27/1999	Version 1.0 release.

Contributors

John Howard	Intel Corporation
Brad Hosler	Intel Corporation
Steve McGowan	Intel Corporation

Please send comments via electronic mail to ccscomments@usb.org

Table of Contents

1. Introduction

The following feature specification describes a standard format for CCS devices to send messages to the CCS driver. The standardized format allows easy parsing of messages by the CCS driver for a variety of CCS functions. Notifications can be used to indicate to the host events, changes in state or other characteristics related to CCS features.

1.1 Purpose

The purpose of this document is to provide a common specification for host notification of asynchronous CCS related events.

1.2 Scope

This document fully describes the Common Class Notification Feature extension for USB Devices. It describes:

- The CCS notification interface.
- The format of dynamic notification requests.
- The class code used by the notification interface.
- How to identify the notification types supported by a device.

1.3 Related Documents

USB Specification, Version 1.1, available at http://www.usb.org.

USB Common Class Specification, Version 1.0, available at http://www.usb.org.

CCS Feature Specification: Version Descriptor, available at http://www.usb.org.

2. Management Overview

It is assumed that a variety of features supported by USB devices will require notifying the host of related asynchronous events that occur on the device. Each notification generates a message over an interrupt pipe. To minimize the hardware requirements, messages sent by features share a single interrupt pipe. These assumptions require that a method be defined to multiplex the messages over the shared interrupt pipe. Common Class Notifications provide a standard method for multiple features to signal the host of asynchronous events.

For example, infrared control interfaces (e.g. IrDA-C) are designed to allow devices such as infrared keyboards, mice, joysticks, and other input/output devices to communicate within a typical room in a home. CCS Notifications can be used to notify the host of the discovery or removal of an IrDA-C device.

The USB Common Class allows Features to be added as new functionality is identified. At any point in time, the latest “version” of CCS is the latest revisions of currently approved CCS Features. When a system software vendor develops a CCS driver they will implement all Features approved to date. If system software encounters a device that supports a Feature defined after the release of its CCS drivers then it is important that the maximum functionality of the device be provided and that the device does not break the drivers. By defining a common message format for notifications, system software can easily ignore messages that it does not understand and handle the ones that it does.

3. Functional Characteristics

3.1 Requirements

CCS Notification support is required to exist on both the Host and Device.

A device that requires CCS notification must also provide a Common Class Notification pipe to inform the host of changes in the state of the device.

All features on a device that use Notifications share a single interrupt pipe.

A common message format allows received Notifications to be easily routed to the appropriate handler.

The CCS driver must be able to identify the CCS feature set supported by the device.

Notifications are transmitted over the interrupt pipe declared in an Endpoint Descriptor under the CCS Interface Descriptor.

3.2 Specification

A device that requires CCS notification support will declare a CCS class interface with an Interrupt In endpoint to provide the notification pipe. The host will bind the USB system software to the CCS interface at device enumeration time. The CCS driver will recognize notification requests and route them to the appropriate handler.

The specification for notifications includes the following:

- A CCS notification message format.
- Definition of a “notification pipe”.
- A CCS class code to identify the interface of the notification pipe.
- A method of identifying the Features supported by a device.

Each of these areas is discussed in more detail below.

4. Operational Model

4.1 Managing Notifications

This section provides an overview of the Common Class Notification mechanism, the definition of the format for Notification messages and the method of defining a notification pipe.

In order to support this feature the device must contain a CCS interface with a Class Code as defined by the USB-IF. The Interrupt In endpoint associated with a CCS interface will be used to transmit the notification requests to the host.

If a capability change, state change, or other event of interest occurs in a device, the device can send a Notification to the host.

The host will route Notifications to the appropriate handler. The CCS feature handler will then perform the operations appropriate for the notification.

A single notification endpoint can carry notification messages from multiple sources.

4.2 Feature Set Identification

The USB Common Class Version descriptor will be used to identify the Features required by the device, including the Notification Feature.

5. Descriptors

This section defines the feature-specific descriptors for the notification feature.

5.1 Interface Descriptor

CCS Notification support takes place at the interface level. To invoke Common Class Features a CCS Interface Descriptor must be declared. See the CCS Logical-Device Feature Specification for a description of the CCS Interface Descriptor.

5.2 Version Descriptor

A device that employs CCS Notification functionality must declare a CCS Version Descriptor. The CCS Version descriptor will contain a Version Information Record for the CCS Notification feature.

The Feature ID (*bFeatureID*) for the Shared Endpoint feature is 0x02.

All Feature Flags (*bmFeatureFlags*) are reserved and set to 0.

See the CCS Version Descriptor Feature Specification for a detailed description of the CCS Version Descriptor.

5.3 Endpoint Descriptor

CCS notifications require the declaration of an "Interrupt In" endpoint descriptor.

Below is an example of an endpoint descriptor that would be used to declare the CCS interrupt in endpoint.

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes (0x07)
1	*bDescriptorType*	1	Constant	ENDPOINT Descriptor Type
2	*bmAttributes*	1	Bit Map	This field describes the endpoint's attributes when it is configured using the *bConfigurationValue*. For a CCS notification bits 0:1 will always be 11, indicating an Interrupt endpoint. All other bits are reserved
3	*wMaxPacketSize*	2	Number	The maximum packet size.
5	*bInterval*	1	Number	Interval for polling endpoint for data transfers. Expressed in milliseconds. The value of this field depends on the worst case requirements of the feature.
6	*bEndpointAddress*	1	Endpoint	The address of the endpoint on the USB device described by this descriptor. The address is encoded as follows: Bit 0..3: The endpoint number Bit 4..6: Reserved, reset to 0 Bit 7: Direction = 1 to define a an interrupt in IN endpoint.

Table 5. Descriptors-1: CCS Notification Endpoint Descriptor

5.4 Notification Format

All CCS notifications requests are made using interrupt in transfers. The notification and the notification's parameters are sent to the device in the notification packet. The device is responsible for establishing the values passed in the following fields. Every notification has eight or more bytes, used as follows:

Offset	Field	Size	Value	Description
0	*bmNotificationType*	1	Bit-map	Characteristics of notification D7 Reserved, must be 1 D6..5 Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved D4..0 Originator 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4..31 = Reserved
1	*bNotification*	1	Value	Feature specific notification ID
2	*wValue*	2	Value	Word-sized field that varies according to notification
4	*wIndex*	2	Value	Word sized field that varies according to notification - typically used to pass a logical-device, interface or endpoint number
6	*wLength*	2	Count	Number of bytes to transfer following this notification message

Table 5. Descriptors-2: CCS Notification Format

5.4.1 bmNotificationType

This bit-mapped field identifies the characteristics of the specific notification.

CCS Feature specifications, Class specifications, or device vendors may specify notifications.

Notifications originate from the physical device, a logical-device, an interface on the device, or a specific endpoint on a device. When a logical-device, interface or endpoint is specified, the *wIndex* field identifies the logical-device, interface or endpoint.

5.4.2 bNotification

This field specifies the particular notification. The *Type* bits in the *bmNotificationType* field modify the meaning of this field. Individual feature specifications that require CCS Notification support define values for the *bNotification* field. The *bNotification* value 0 is reserved.

5.4.3 wValue

The contents of this field vary according to the notification. It is used to pass a notification specific parameter to the host.

5.4.4 wIndex

The contents of this field vary according to the request. It is used to pass a notification specific parameter to the host. For notifications originating from logical-devices, interfaces, or endpoints, *wIndex* identifies the logical-device, interface, or endpoint.

5.4.5 wLength

This field specifies the length of the data transferred during the second phase of the transfer.