[MS-USBEPD]:

USB Protocol: Platform Detection Extensions

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](#) or the [Microsoft Community Promise](#). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](#).
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
4/23/2024	1.0	New	Released new document.

Table of Contents

1   Introduction

The USB Protocol: Platform Detection Extensions protocol extends the USB 3.2 protocol to allow USB devices that support Microsoft OS 2.0 Descriptors to query a USB Host's platform to detect what platform they are currently connected to. If the USB device and the Host system that it is connected to support the USB Platform Detection Protocol then the USB device can take advantage of more detailed driver and firmware compatibility and the special features the Host has to offer.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1   Glossary

This document uses the following terms:

device: Any UPnP-enabled device.

driver: Software that allows applications to interact with a hardware device by using abstract/high-level constructs.

universal serial bus (USB): An external bus that supports Plug and Play installation. It allows devices to be connected and disconnected without shutting down or restarting the computer.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2   References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1   Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[MSLEARN-OS20Desc] Microsoft Corporation, "Microsoft OS 2.0 Descriptors Specification", July 2018, https://learn.microsoft.com/en-us/windows-hardware/drivers/usbcon/microsoft-os-2-0-descriptors-specification

Note Accept the License Agreement to download the file.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, https://www.rfc-editor.org/info/rfc2119

[USB-IF-SPC32] USB-IF, "Document Library", Universal Serial Bus 3.2 Specification Revision 1.1 - June 2022, https://www.usb.org/documents

Note Search: USB 3.2, Category: Specification, and Technology: USB 3.2

1.2.2   Informative References

None.

1.3   Overview

The USB Protocol Platform Detection Extensions protocol is based on the USB 3.2 Protocol [USB-IF-SPC32] and Microsoft OS 2.0 Descriptors protocol [MSLEARN-OS20Desc]. It is a protocol that allows **universal serial bus (USB) devices** to query a USB Host's platform to detect what platform they are currently connected to. This is to provide more detailed **driver** and firmware compatibility for various USB Hosts that the USB device can be used with.<1>

This protocol also allows USB devices to detect if they are not on a specific platform. If that platform is known to implement the USB Protocol Extensions for Platform Detection and the Host system that the USB device is connected to does not implement the USB Platform Detection Protocol, then the Host is not one of the known systems, instead it is a different unknown system.

A USB input device, for example, can have a custom driver on several operating systems that allows it take advantage of certain features not included in the standard USB input protocols that are supported by all USB Hosts. Such a device can use the USB Platform Detection Extensions protocol to determine which Host USB platform the device is connected to. Based on the result the USB device can alter its behavior to either expect a custom driver or to behave as a standard USB device for which the OS does not provide a driver.

1.4   Relationship to Other Protocols

The USB Protocol: Platform Detection Extensions protocol is an extension of the existing USB 3.2 Protocol Specification ([USB-IF-SPC32]) and Microsoft OS 2.0 Descriptor Specification ([MSLEARN-OS20Desc]). This protocol uses the OS 2.0 Descriptor's driver Compatible ID to opt into support for the USB Platform Detection Extension by a USB device. This protocol also makes extensive use of USB channel Control Transfer outlined in the USB 3.2 Specification. Therefore, basic familiarity is assumed with the USB 3.2 Protocol Specification and the Microsoft OS Descriptor 2.0 specification.

1.5   Prerequisites/Preconditions

The Platform Detection extensions to the USB 3.2 Protocol require that the **USB device** fully enumerate with the USB Host, for more details about this process please see the USB 3.2 specification [USB-IF-SPC32] enumeration section 9.1.2.

This protocol is also required to implement the Microsoft OS Descriptors 2.0 protocol, and to specify a Compatible ID of "PLATDET". For more details about this process please see the Microsoft OS 2.0 Descriptor specification [MSLEARN-OS20Desc] Compatible ID descriptor subsection.

1.6   Applicability Statement

This protocol is only intended to be used by **USB devices** connecting to an unknown USB Host for which they wish to detect what is the specific operating system or platform Host.

This protocol is not intended to be a replacement for USB device functionality or re-define any aspect of the Microsoft OS Descriptors 2.0 protocol ([MSLEARN-OS20Desc]) for which this protocol extends. Rather it intends to define a specific Compatible ID and its functions to allow a USB device to detect its Host's platform.

1.7   Versioning and Capability Negotiation

None.

1.8   Vendor-Extensible Fields

None.

1.9   Standards Assignments

None.

2 Messages

2.1 Transport

USB Protocol: Platform Detection Extensions protocol uses the USB 3.2 Specifications' **USB** control channel exclusively for transport of the protocol specific messages ([USB-IF-SPC32]). The USB vendor specific control request number is 0xE0 for all messages.

2.2 Message Syntax

The following sections define the composition of USB messages used by the USB Protocol: Platform Detection Extensions protocol.

2.2.1 Common Message Header

All **USB** control pipe messages use the following common header.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Status								Command																Connection ID							
...								Sequence Number																Payload (variable)							
...																															
...																															
...																															

Status (1 byte): An 8-bit valid USB transaction status.

Command (2 bytes): A 16-bit Command ID, specified by this protocol.

Connection ID (2 bytes): A 16-bit ID assigned by the Host for the duration of a USB enumeration session to uniquely identify specific platform detection negotiations.

Sequence Number (2 bytes): A 16-bit unsigned integer that represents the number of times a specific command message has been sent. The Host and Device MUST increment this number from an initial value of one and the Host and Device MUST overflow from 0xFFFF to 0x0001 without passing through zero (0x0000).

Payload (variable): Message specific data if any for the associated **Command**.

Bit Numbering

In this protocol all multibyte values are sent as little-endian even on big-endian Host or Device architectures. Implementers should take care to ensure that they decode any messages over the wire into the appropriate endianness for their target platform.

2.2.2 Device Registration Message

The Device Registration message is a send-receive-reply message sent by the Host after enumeration. The Host MUST send this message within 800ms of enumeration, otherwise the device can assume that the Host does not support the platform detection protocol.

Message 1 of 2: Host to Device Registration Message

The initial registration message from the Host to the device. This message lets the device know it wants to do platform detection and when ACK'd by the device confirms to the Host that the platform can now be sent.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Status								Command																Connection ID							
...								Sequence Number															B								

Status (1 byte): A valid USB transaction status. The Host MUST specify ACK (0x1) to acknowledge**.**

Command (2 bytes): The 16-bit device registration Command ID, the Host MUST specify 0x0001.

Connection ID (2 bytes): A 16-bit ID assigned by the Host for the duration of a USB enumeration session to uniquely identify specific platform detection negotiations.

Sequence Number (2 bytes): A 16-bit unsigned integer that represents the number of times a specific command message has been sent. The Host and Device MUST increment this number from an initial value of one and the Host and Device MUST overflow from 0xFFFF to 0x0001 without passing through zero (0x0000).

B - Payload (0): Empty, zero bytes. Followed by EOM: End of Message.

Message 2 of 2: Device to Host Registration Message

The Device MUST send this message with 500ms of receiving the device registration message from the Host, otherwise the Host can assume the Device is defective and terminate power to the Device's USB Port.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Status								Command																Connection ID							
...								Sequence Number															B								

Status (1 byte): A valid USB transaction status. Usually ACK (0x01) or NAK (0x00) a NAK is a (Negative Acknowledgement) failure, and the message will be retried as per the USB 3.2 specification, [USB-IF-SPC32].

Command (2 bytes): The 16-bit device registration Command ID, the device MUST specify 0x0001.

Connection ID (2 bytes): A 16-bit ID assigned by the Host for the duration of a USB enumeration session to uniquely identify specific platform detection negotiations. The Device MUST use the Connection ID specified by the Host in the preceding registration message.

Sequence_Number (2 bytes): A 16-bit unsigned integer that represents the number of times a specific command message has been sent. The Host and Device MUST increment this number from

an initial value of one and the Host and Device MUST overflow from 0xFFFF to 0x0001 without passing through zero (0x0000).

B - Payload (0): Empty, zero bytes. Followed by EOM: End of Message.

2.2.3 Platform Information Message

The Platform Information Message is a send-receive-reply message sent by the Host. The Host MUST send this message after Device Registration messages have been ACK'd, by the **device**.

Message 1 of 2: Host to Device Platform Information Message

The Platform Information message from the Host to the device. This message provides the platform ID to the device to allow a USB device to detect its Host's platform.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Status								Command																Connection ID							
...								Sequence Number																Payload							
...																															

Status (1 byte): A valid **USB** transaction status. In this case the Host MUST specify ACK (0x1).

Command (2 bytes): The 16-bit device registration Command ID, the Host MUST specify 0x0002.

Connection ID (2 bytes): A 16-bit ID assigned by the Host for the duration of a USB enumeration session to uniquely identify specific platform detection negotiations.

Sequence Number (2 bytes): A 16-bit unsigned integer that represents the number of times a specific command message has been sent. The Host and Device MUST increment this number from an initial value of one and the Host and Device MUST overflow from 0xFFFF to 0x0001 without passing through zero (0x0000).

Payload (2 bytes): A 16-bit platform ID<2> in little-endian format defined in the following table. Followed by EOM: End of Message.

Platform ID	Meaning
0x0000	Reserved, this value MUST NOT be used by either Host or Device for any reason.
0x0001	Windows 10 operating system
0x0002	Windows 11 operating system and later
0x0003	Windows 10 IOT Core
0x0004	Windows 11 IOT or later
0x0005	Windows Server 2016 operating system, Windows Server 2019 operating system, or Windows Server 2022 operating system
0x0006	Windows Server 2025 operating system and later
0x0007	Xbox console, Xbox One and later
0x0008	Microsoft OneCore based operating system

Platform ID	Meaning
0x0009	Other operating systems not better described by one of the above IDs
0x000A-0xFFFF	Reserved for future expansion, the Host MUST NOT send this platform value.

Message 2 of 2: Device to Host Platform Information Message

The Device MUST send this message with 500ms of receiving the Host's Platform Information message, otherwise the Host can assume the Device is defective and terminate power to the Device's USB Port.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Status								Command																Connection ID							
…								Sequence Number																B							

Status (1 byte): A valid USB transaction status. Usually ACK (0x01) or NAK (0x00) a NAK is a (negative acknowledgement) failure, and the message will be retried as per the USB 3.2 specification.

Command (2 bytes): The 16-bit device registration Command ID, the Host MUST specify 0x0002.

Connection_ID (2 bytes): A 16-bit ID assigned by the Host for the duration of a USB enumeration session to uniquely identify specific platform detection negotiations. The Device MUST use the **Connection ID** specified by the Host in the preceding registration message.

Sequence Number (2 bytes): A 16-bit unsigned integer that represents the number of times a specific command message has been sent. The Host and Device MUST increment this number from an initial value of one and the Host and Device MUST overflow from 0xFFFF to 0x0001 without passing through zero (0x0000).

B – Payload (0): Empty, zero bytes. Followed by EOM: End of Message.

3   Protocol Details

3.1   Host and Device Details

The **USB** platform detection protocol re-uses the specified roles of USB Host and USB Device from the USB 3.2 Specification, [USB-IF-SPC32].

Host Details

The Host follows the roles of the USB Host as specified in the USB 3.2 specification. In the platform detection extensions, the Host initiates both USB control channel messages with send-receive-reply message semantics.

Device Details

The Device follows the roles of the USB Device as specified in the USB 3.2 specification. In the platform detection extensions, the Device replies to both USB control channel messages with send-receive-reply message semantics.

3.1.1   Abstract Data Model

None.

3.1.2   Timers

None.

3.1.3   Initialization

None.

3.1.4   Higher-Layer Triggered Events

None.

3.1.5   Message Processing Events and Sequencing Rules

None.

3.1.6   Timer Events

None.

3.1.7   Other Local Events

None.

4   Protocol Examples

Consider the example of a Contoso USB device that would like to load a special driver that takes advantage of unique features only present on Windows Server 2016 operating system and later.

First the Device will enumerate with the Host as described in the USB 3.2 specification ([USB-IF-SPC32]), after enumeration the Host can request a device Compatible ID via the Microsoft OS 2.0 Descriptors protocol ([MSLEARN-OS20Desc]). In our example the Contoso **USB device** specified the "PLATDET" USB Compatible ID from the Host.

As a result, the Windows Server 2016 and later Host sends a platform detection registration message to the device to confirm that the Host does indeed support USB platform detections.

When the Contoso USB device does not receive a reply in the timeframe specified in this protocol it assumes the platform is not one that supports USB platform detection, and the device can choose to renumerate, or exercise normal functionality as specified in the USB 3.2 specification.

However, if the Contoso USB device does receive a registration message within the timeframe, then it sends a valid reply with the ACK (0x1) status set meaning it acknowledges the receipt of the registration and it does support USB platform detection.

The Host will then continue to send Platform Information messages to the device until the device sends an ACK status reply at which point no further platform detection messages are sent. If the platform specified is Windows Server 2016 or later, the device will re-enumerate with a special USB descriptor that makes use of Windows Server specific features.

Otherwise, the device will renumerate in its default USB mode suitable for all non-Windows server platforms.

5   Security

5.1   Security Considerations for Implementers

Implementers of the USB Platform Detection extensions protocol should take care to validate the size of the USB control transactions to ensure they meet or exceed the specified messages' size in this protocol.

Note that per the USB 3.2 specification USB Hosts or devices can add any number of bytes to the end of a control transfer message, as a result implementations MUST not fail if they receive too much data, only too little data.

5.2   Index of Security Parameters

None.

6   Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

The terms "earlier" and "later", when used with a product version, refer to either all preceding versions or all subsequent versions, respectively. The term "through" refers to the inclusive range of versions. Applicable Microsoft products are listed chronologically in this section.

Windows Client

▪   Windows 11, version 24H2 operating system

Windows Server

▪   Windows Server 2025 operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

<1> Section 1.3: This is an in-box Windows component WDM driver that sends USB control pipe messages to implement this protocol. It may include Windows Desktop, Windows Server, Xbox or even non-Microsoft operating systems that also implement both the USB Platform Detection Protocol and the Microsoft OS Descriptors.

<2> Section 2.2.3: Other platforms can contact Microsoft to obtain a valid platform ID in the USB Platform Detection Protocol ID namespace, please contact dochelp@microsoft.com.

7   Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

A

Applicability 5

C

Capability negotiation 5
Change tracking 15
Common Message Header message 7

D

Device Registration Message message 8

F

Fields - vendor-extensible 5

G

Glossary 4

I

Implementer - security considerations 13
Index of security parameters 13
Informative references 4
Introduction 4

M

Messages
    Common Message Header 7
    Device Registration Message 8
    Platform Information Message 9
    transport 7

N

Normative references 4

O

Overview (synopsis) 5

P

Parameters - security index 13
Platform Information Message message 9
Preconditions 5
Prerequisites 5
Product behavior 14

R

References 4
    informative 4
    normative 4
Relationship to other protocols 5

S

Security
    implementer considerations 13
    parameter index 13
Standards assignments 6

T

Tracking changes 15
Transport 7

V

Vendor-extensible fields 5
Versioning 5