Document Identifier: DSP0283

Date: 2024-05-21

Version: 1.0.1

Management Component Transport Protocol (MCTP) Universal Serial Bus (USB) Transport Binding Specification

Supersedes: 1.0.0

Document Class: Normative

Document Status: Published

Document Language: en-US

CONTENTS

Figures

Tables

Foreword

The Management Component Transport Protocol (MCTP) Universal Serial Bus (USB) Transport Binding Specification (DSP0283) was prepared by the PMCI working group.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about DMTF, see https://www.dmtf.org.

USB Implementers Forum, Inc. is a non-profit corporation founded by the group of companies that developed the Universal Serial Bus specification. The USB-IF was formed to provide a support organization and forum for the advancement and adoption of Universal Serial Bus technology. For information about USB organization see https://www.usb.org.

Acknowledgments

DMTF acknowledges the following individuals for their contributions to this document:

Editor:

- Yuval Itkin – NVIDIA Corporation

Contributors:

- Patrick Caporale – Lenovo
- Samer El-Haj-Mahmoud – ARM Inc
- Michael Garner – Meta
- Jeff Hilland – Hewlett Packard Enterprise
- Eliel Louzoun – Intel Corporation
- Chandra Nelogal – Dell Technologies
- Edward Newman – Hewlett Packard Enterprise
- Stevens, Pierre-Philippe – Advanced Micro Devices
- Paul Sack – Marvell Ltd
- Hemal Shah – Broadcom Inc.
- Bob Stevens – Dell Technologies

Introduction

The Management Component Transport Protocol (MCTP) Universal Serial Bus (USB) transport binding defines a transport binding for facilitating MCTP communication between platform management system components (e.g. management controllers, management devices) over USB 2.0.

The *MCTP Base Specification* describes the protocol and commands used for communication within and initialization of an MCTP network. The Management Component Transport Protocol (MCTP) Universal Serial Bus (USB) transport binding definition in this specification includes a packet format, USB endpoint descriptors, message routing, and discovery mechanisms for MCTP over USB 2.0 communications.

Management Component Transport Protocol (MCTP) Universal Serial Bus (USB) Transport Binding Specification

1 Scope

This document provides the specification for the Management Component Transport Protocol (MCTP) transport binding using USB.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

DMTF DSP0004, Common Information Model (CIM) Metamodel 3.0, https://www.dmtf.org/standards/published_documents/DSP0004_3.0.pdf

DMTF DSP0222, Network Controller Sideband Interface (NC-SI) Specification 1.1 https://www.dmtf.org/sites/default/files/standards/documents/DSP0222_1.1.pdf

DMTF DSP0223, Generic Operations 1.0, https://www.dmtf.org/standards/published_documents/DSP0223_1.0.pdf

DMTF DSP0236, Management Component Transport Protocol (MCTP) Base Specification 1.3 https://www.dmtf.org/sites/default/files/standards/documents/DSP0236_1.3.pdf

DMTF DSP0239, Management Component Transport Protocol (MCTP) IDs and Codes 1.8 https://www.dmtf.org/sites/default/files/standards/documents/DSP0239_1.8.pdf

DMTF DSP0256, Management Component Transport Protocol (MCTP) Host Interface Specification 1.0 https://www.dmtf.org/sites/default/files/standards/documents/DSP0256_1.0.pdf

DMTF DSP1001, Management Profile Specification Usage Guide 1.1, https://www.dmtf.org/standards/published_documents/DSP1001_1.1.pdf

ISO/IEC Directives, Part 2, Principles and rules for the structure and drafting of ISO and IEC documents, https://www.iso.org/sites/directives/current/part2/index.xhtml

USB Implementers Forum, Inc. Universal Serial Bus Specification version 2.0 https://www.usb.org/document-library/usb-20-specification

3 Terms and definitions

In this document, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause.

The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"), "may", "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 7. The terms in parentheses are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Clause 7 specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 6.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The terms defined in DSP0004, DSP0223, and DSP1001 apply to this document. The following additional terms are used in this document.

3.1

MCTP USB Endpoint

a USB interface on which MCTP over USB communication is supported

4 Symbols and abbreviated terms

The abbreviations defined in DSP0004, DSP0223, and DSP1001 apply to this document. The following additional abbreviations are used in this document.

4.1

USB

Universal Serial Bus

5 Conventions

The conventions described in the following clauses apply to this specification.

5.1 Reserved and Unassigned Values

Unless otherwise specified, any reserved, unspecified, or unassigned values in enumerations or other numeric ranges are reserved for future definition by DMTF.

Unless otherwise specified, numeric or bit fields that are designated as reserved shall be written as 0 (zero) and ignored when read.

5.2 Byte Ordering

Unless otherwise specified, byte ordering of multi-byte numeric fields or bit fields is "Big Endian" (that is, the lower byte offset holds the most significant byte, and higher offsets hold lesser significant bytes).

6 MCTP over USB Transport

This document defines the medium-specific transport binding for transferring MCTP packets between endpoints on USB using USB Bulk endpoints.

A MCTP over USB compliant USB device shall support MCTP over USB communications on at least one USB interface of the device. If a MCTP over USB compliant USB device supports MCTP over USB communications on more than one USB interface, then MCTP over USB communication on each such USB interface shall be independent from MCTP over USB communications on other USB interfaces.

6.1 MCTP use with USB

6.1.1 USB bus physical topology

The physical topology of the USB bus is presented in Figure 1. There is a single host device that operates as the USB tree Root (typically it is a Management Controller, Embedded Controller, etc.) and there may be multiple devices sharing the same USB bus tree. A set of USB hubs may be used to enable connection of multiple USB devices to the same USB host device.

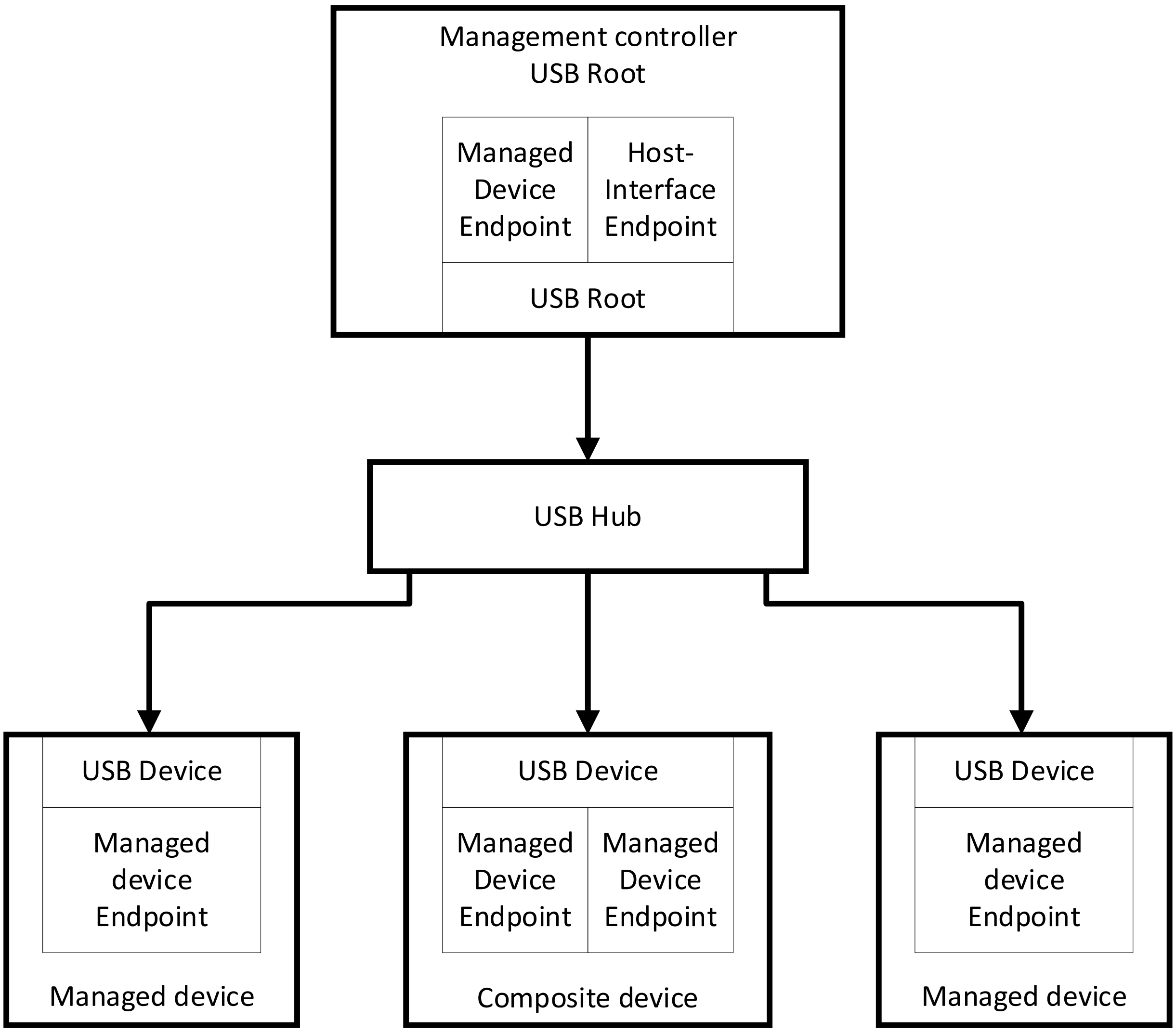

Figure 1 – Physical topology of USB bus

6.1.2 MCTP bus owner using USB bus

The MCTP Bus Owner is the USB Root. It is responsible for the discovery and managing the EID assignments for the MCTP endpoints on the USB bus. The discovery of the MCTP endpoints is done using the provided USB descriptors of each device that has MCTP interface(s) as part of the device discovery on the USB bus, as detailed in 6.1.4.

The USB host may also be separated from the USB root device. In such a case the USB root is controlled by a separate interface as shown in Figure 2.

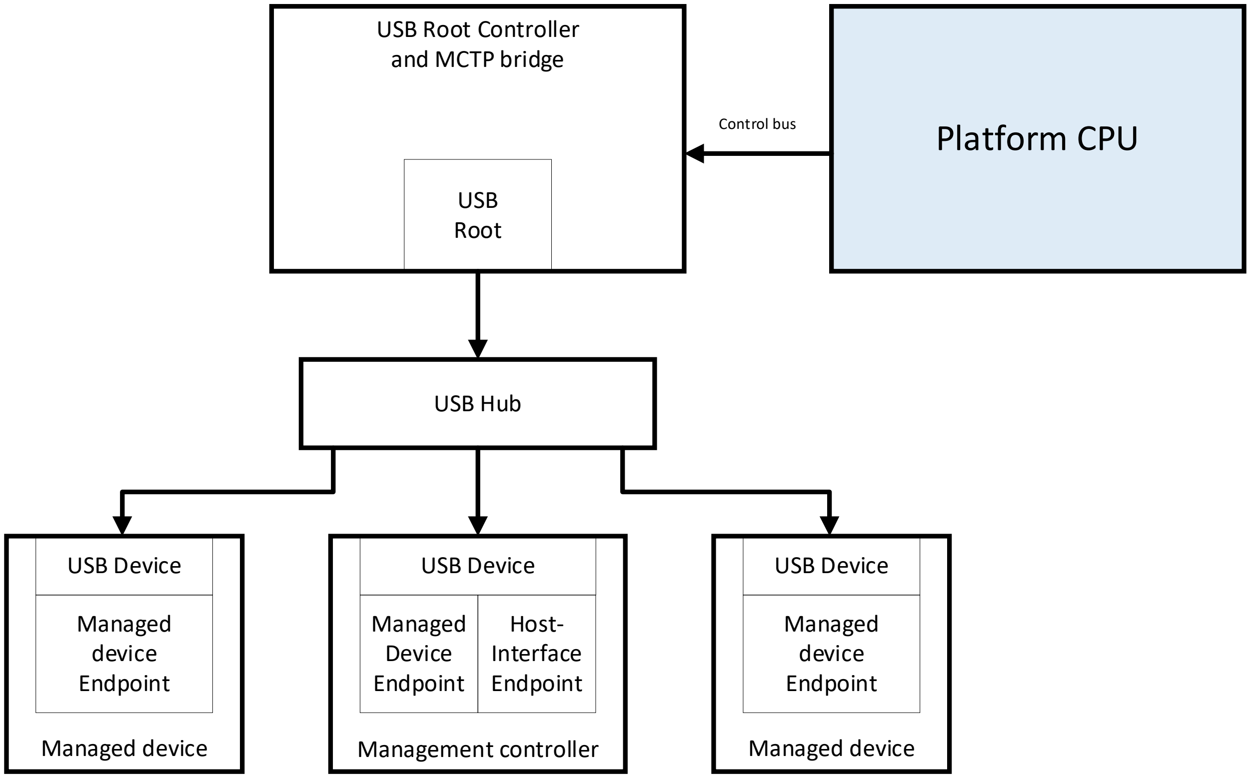

Figure 2 – Separated USB host and USB Root devices

6.1.3 MCTP bridges over USB

The USB root may act as an MCTP bridge. As USB protocol does not allow direct peer-to-peer communication between MCTP endpoints on USB, the USB Root will typically serve as an MCTP bridge for all the MCTP endpoints connected to the same USB Root as shown in Figure 3.

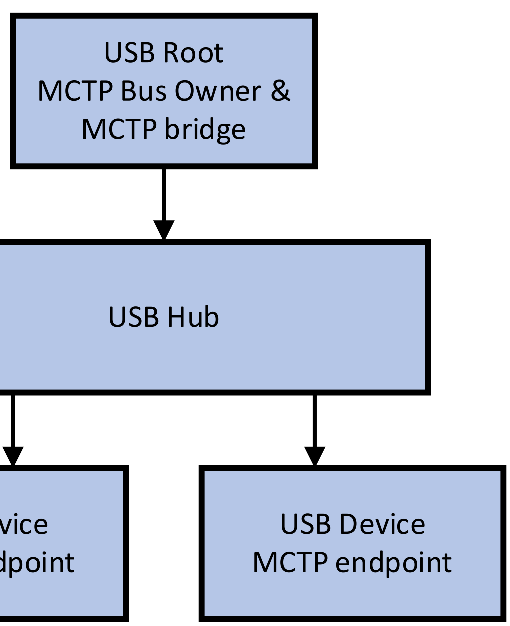

Figure 3 – A USB Root as MCTP bus owner and MCTP bridge

A USB MCTP endpoint can also serve as an MCTP bridge to another MCTP bus as shown in Figure 4.

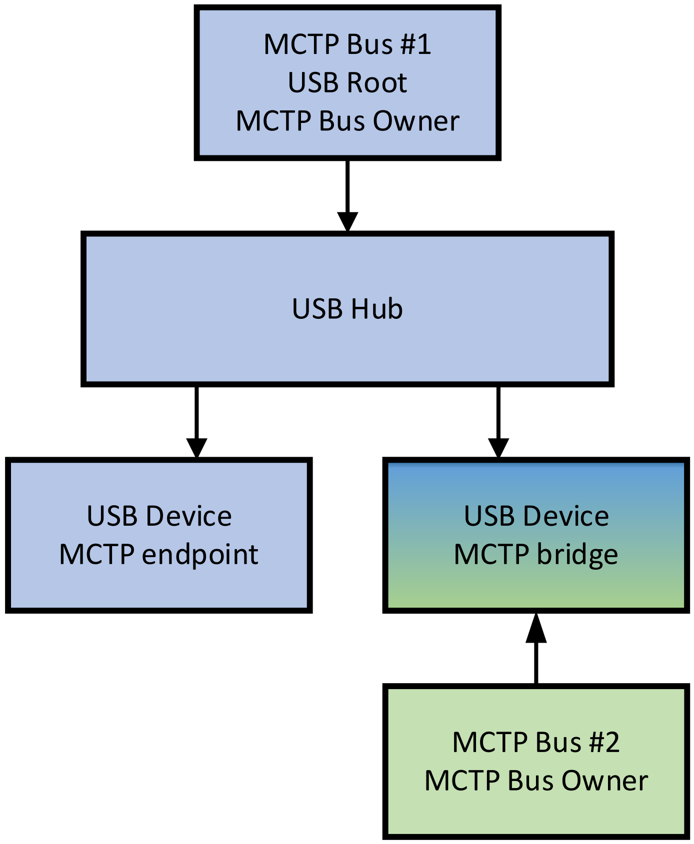

Figure 4 – An MCTP over USB endpoint as an MCTP bridge

6.1.4 Descriptors structure for MCTP endpoint for MCTP over USB

An MCTP over USB endpoint is composed of 2 USB Bulk endpoints:

- Out Bulk endpoint – used to send data from the USB root to the USB device
- In Bulk endpoint – used to send data from the USB device to the USB root

The set of these 2 endpoints is defined as a single USB MCTP interface which is declared by the following USB descriptors. A device may have more than one MCTP endpoint. Each such MCTP endpoint is an independent USB interface.

MCTP over USB is operating in high-speed mode, the endpoint buffer size shall be set to 512 Bytes.

6.1.4.1 Interface descriptor

For every MCTP endpoint there is a single interface descriptor. This USB interface descriptor defines

- Class code – A value of 0x14 defines an MCTP endpoint class
- Sub-Class code
  - A value of 0x0 defines a Management-controller and Managed-Device endpoints
  - A value of 0x1 defines an MCTP Host-Interface endpoint
- Number of endpoints on the USB MCTP endpoint interface, shall be set to 2

- Protocol – Class-specific protocol as follows
    - A value of 0x1 defines MCTP 1.x protocol
    - A value of 0x2 defines MCTP 2.x protocol
    - Other values are reserved
- Alternate settings – shall always be set to 0. An MCTP over USB endpoint shall have no alternate settings.

6.1.4.2 Endpoint descriptor

A descriptor is required for every USB Bulk endpoint. Given that there are 2 USB Bulk endpoints for every MCTP interface there are 2 Bulk endpoint descriptors.

The 2 Bulk endpoints should use the same USB endpoint number.

6.1.4.2.1 Out Bulk endpoint descriptor

This descriptor declares the USB Bulk endpoint that is used to send data from the USB Root to the USB Device. The following attributes shall be defined in this descriptor:

- bEndpointAddress – set to the following 8 bits value:
  [7:4] - 0000,
  [3:0] - Bulk_Endpoint_Number_In_USB_Device
- bmAttributes – Set to 0x02 to declare a Bulk endpoint
- wMaxPacketSize
    - Set to 512, declaring a 512 Bytes buffer size
- bInterval – set to 0x01
    - High-speed devices, declaring that the host shall not try to access the endpoint again during the same micro-frame after receiving a NAK response

      Using this setting minimizes the system idle power by lowering the maximal NAK rate on every USB endpoint to 8000 times per second. This sets the maximal additional response latency in such a case to 125µsec

      Implementation note: While USB specification defines bInterval as a method for setting the maximal NAK rate, there are implementations which may not lower the polling rate based on this parameter.

6.1.4.2.2 In Bulk endpoint descriptor

This descriptor declares the USB Bulk endpoint that is used to send data from the USB Device to the USB Root. The following attributes shall be defined in this descriptor

- bEndpointAddress – set to the following 8 bits value:
  [7:4] - 1000,
  [3:0] - Bulk_Endpoint_Number_In_USB_Device
- bmAttributes – Set to 0x02 to declare a Bulk endpoint
- wMaxPacketSize
    - Set to 512, declaring a 512 Bytes buffer size
- bInterval – set to 0x01
    - For high-speed devices, declaring that the host shall not try to access the endpoint again during the same micro-frame after receiving a NAK response

Using this setting minimizes the system idle power by lowering the maximal NAK rate on every USB endpoint to 8000 times per second. This sets the maximal additional response latency in such a case to 125µsec

Implementation note: While USB specification defines bInterval as a method for setting the maximal NAK rate, there are implementations which may not lower the polling rate based on this parameter.

6.2 Packet Format

The use of USB bulk endpoint for MCTP over USB does require adding a medium-specific header for each MCTP packet as shown in Figure 5 – MCTP 1.x over USB packet format below.

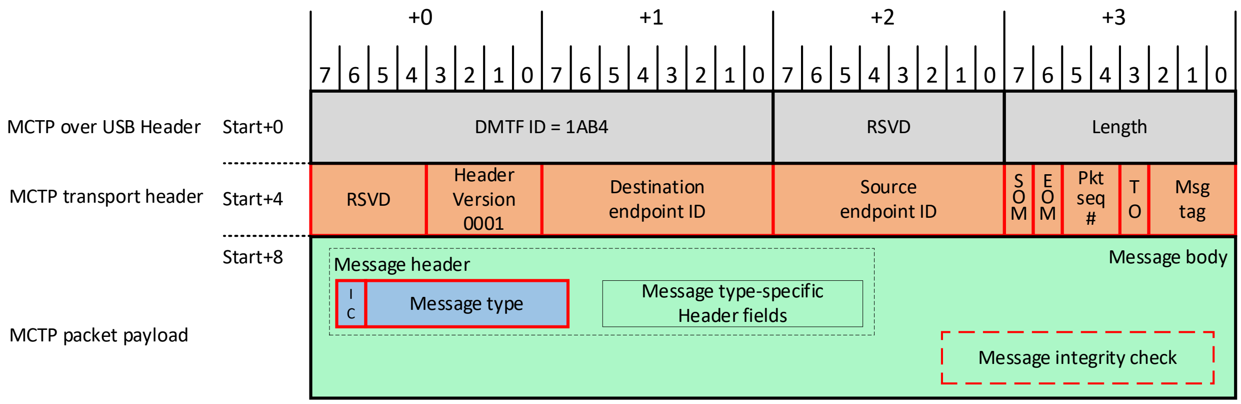

Figure 5 – MCTP 1.x over USB packet format

The fields in the "MCTP over USB Header" are specific to carrying MCTP packets using USB Bulk transfers. The fields labeled "MCTP transport header" and "MCTP packet payload" are common fields for all MCTP packets and messages and are specified in MCTP. This document defines the location of those fields when they are carried in a USB Bulk transfer.

Table 1 lists the MCTP over USB Header fields and values.

Table 1 – MCTP over USB Header Fields

Byte offset	Field	Description
0	DMTF ID	DMTF Identifier. Always set to 0x1AB4, matching DMTF Vendor ID as registered in PCI-Sig.
2	Reserved	MCTP Reserved (8 bits). Shall always be set to 0 when generating a packet. Shall be ignored on receive.
3	Length	Length: Length of the MCTP over USB packet in Bytes, starting from the "MCTP over USB Header" to the last byte in the "MCTP packet payload", implementations shall support the baseline transmission unit defined in DSP0236.

The MCTP packets are sent as the data to the designated USB Bulk endpoint. MCTP traffic over USB shall use High-Speed (480M bits-per-second) mode.

Figure 6 illustrates HS Bulk data transfer. Every transfer starts with a Token which indicates the data transfer direction and the addressed device and endpoint. Following the token, the data packet is transferred (IN or OUT), and after the data packet transfer is complete, a handshake PID is used to ACK/NACK the transfer. The Token and the Data packet include CRC as shown below.

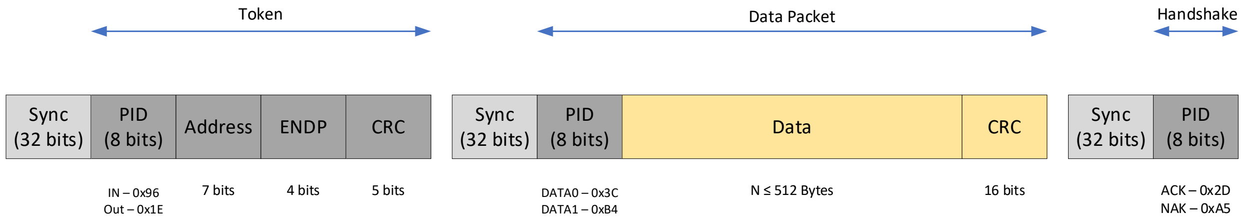

Figure 6 – USB Bulk transfer principal sequence

USB specification does not require data payloads to always be exactly the endpoint buffer size. Therefore, if a data payload is less than the endpoint buffer size, it does not need to be padded to the endpoint buffer size.

The MCTP packet cannot be larger than the endpoint buffer size. The payload of the USB packet contains any combination of one or more MCTP packets destined to or through the same Endpoint-ID (EID). Refer to Figure 7 – USB Packet with single MCTP packet payload, Figure 8 – USB packet with 2 MCTP packets payload.

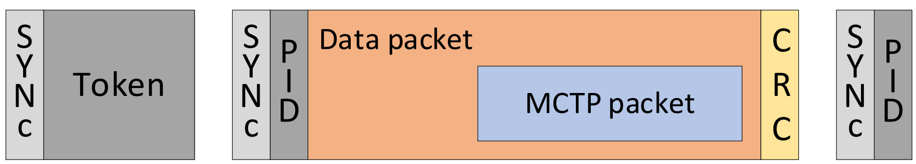

Figure 7 – USB Packet with single MCTP packet payload

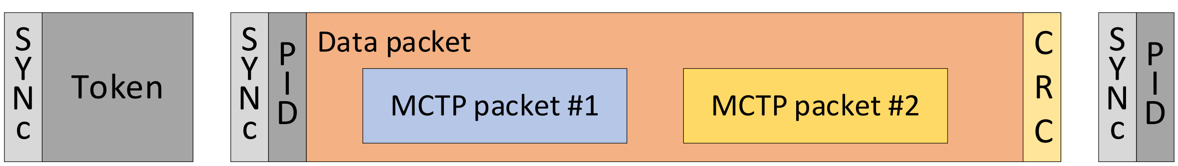

Figure 8 – USB packet with 2 MCTP packets payload

6.3 Error handling

USB Bulk data transfers reliability is ensured in hardware level using error detection and by invoking a limited number of retries. If the retry count is exceeded, the interface shall be reset using a method that is out of scope for this specification.

6.4 MCTP support and capabilities discovery

An MCTP-capable MCTP over USB bus-owner, shall discover all the MCTP capable interfaces on the USB fabric as described below.

6.4.1 Full Endpoint Discovery/Enumeration

The following process is typically used when the MCTP bus owner wishes to discover and enumerate all MCTP endpoints on the USB bus.

1) MCTP-capable devices are identified by their USB descriptors as defined in section 6.1.4. During USB detection and enumeration phase.

2) Following its USB enumeration, an MCTP-capable device shall send the *Discovery Notify* MCTP message , to request EID assignment. A USB interface of a composite device with more than one MCTP endpoint shall send the *Discovery Notify* MCTP message for every MCTP endpoint separately.

3) The MCTP bus owner issues a Prepare for Endpoint Discovery message for every MCTP-capable device using the Broadcast EID as the destination EID. When addressing a composite device with more than one MCTP endpoint, the MCTP bus owner shall issue a Prepare for Endpoint Discovery message for every MCTP-endpoint on that MCTP-capable device using the Broadcast EID as the destination EID.
This message causes each discoverable endpoint on the bus to set its USB endpoint Discovered flag to undiscovered.

4) All MCTP-capable devices that have their Discovered flag set to undiscovered will respond with an Endpoint Discovery response message.

5) The MCTP bus owner should wait for at least MT2 time interval to receive the response. This helps ensure, that all endpoints that received the Prepare for Endpoint Discovery request have processed the request.

6) The MCTP bus owner issues an Endpoint Discovery request message for every MCTP endpoint on an MCTP-capable device using the Broadcast EID as the destination EID. When addressing a composite device with more than one MCTP endpoint, the MCTP bus owner shall issue an Endpoint Discovery message for every MCTP-capable interface using the Broadcast EID as the destination EID.

7) For each response message received from an undiscovered MCTP interface of an MCTP-capable USB device, the MCTP bus owner issues a Set Endpoint ID command to the physical address for the endpoint. This causes the endpoint to set its Discovered flag to *discovered*. From this point, the endpoint shall not respond to the Endpoint Discovery command until another Prepare for Endpoint Discovery command is received, or some other condition causes the Discovered flag to be set back to *undiscovered*.

8) If the MCTP bus owner received any responses to the Endpoint Discovery request issued in Step 6, then it shall repeat steps 6 and 7 until it no longer gets any responses to the Endpoint Discovery request. In this case, then the MCTP bus owner is allowed to send the next Endpoint Discovery request without waiting for MT2 time interval. If no responses were received by the MCTP bus owner to the Endpoint Discovery request within the MT2 time interval, then the discovery process is completed.

After the initial endpoint enumeration, it is recommended that the MCTP bus owner maintains a list of the unique IDs for the endpoints it has discovered and reassigns the same IDs to those endpoints if a USB endpoint number changes during system operation.

Figure 9 provides an example flow of operations for full endpoint discovery.

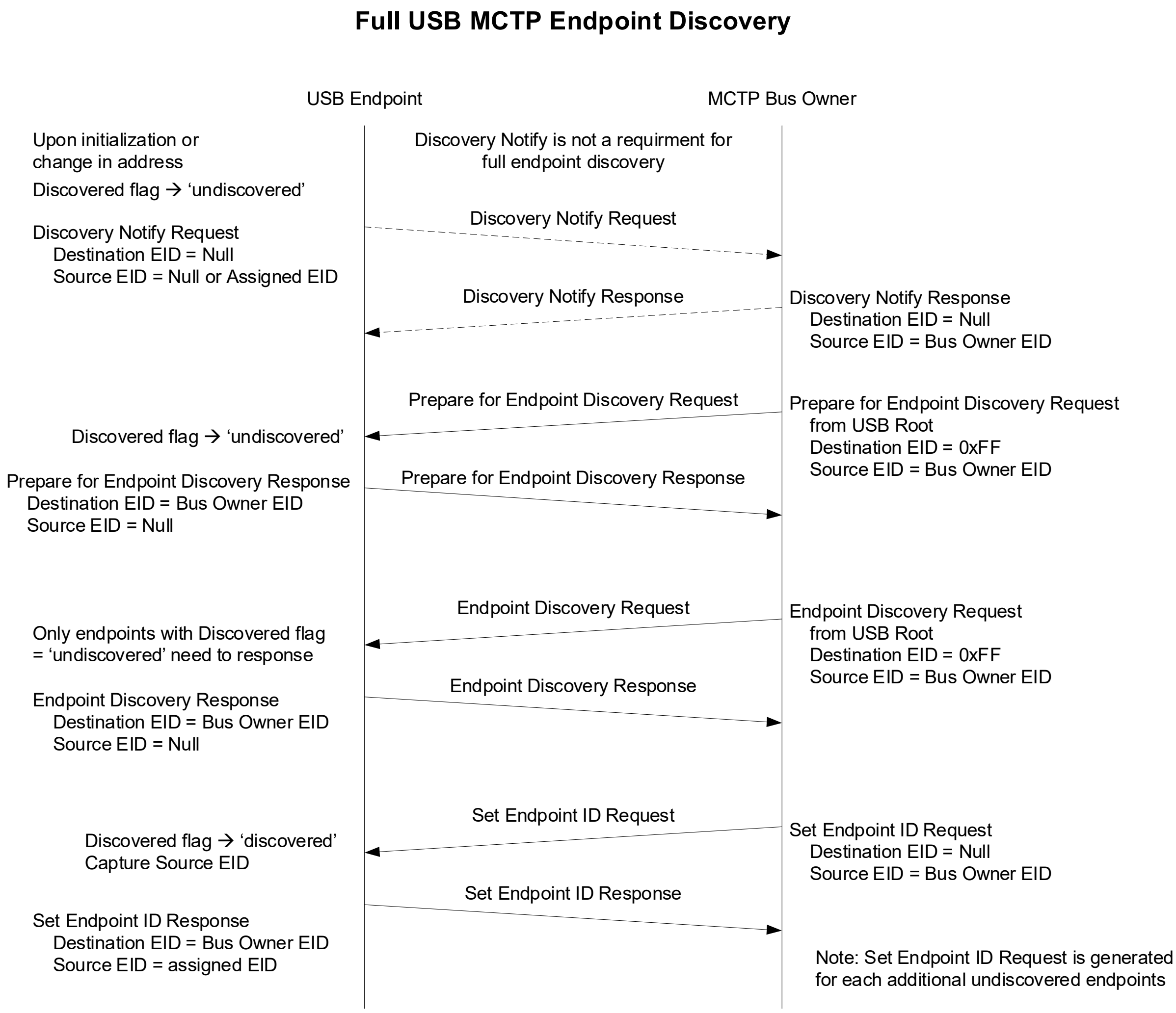

Figure 9 – Flow of Operations for Full MCTP Discovery over USB

6.4.2 Partial Endpoint Discovery/Enumeration

This process is used when the MCTP bus owner wishes to discover endpoints that may have been added to the bus after a full enumeration has been done. This situation can occur if a device has its address change after the full enumeration has been done, or when a hot-plug device is added to the system, or if a device that is already present in the system — but was in a disabled or powered-down state — comes on-line.

The partial discovery process is the same as the full discovery process except that the MCTP bus owner skips the step of broadcasting a Prepare for Endpoint Discovery command in order to avoid clearing the Discovered flags of already discovered endpoints.

The partial discovery process may be initiated when a device that is added or enabled for MCTP sends a Discovery Notify message to the MCTP bus owner. The MCTP bus owner may also elect to periodically issue a broadcast Endpoint Discovery message to test for whether any undiscovered endpoints have been missed. The Discovery Notify message provides the MCTP bus owner with the address/endpoint of the MCTP USB endpoint. The MCTP bus owner can then send a directed Endpoint Discovery message to the endpoint to confirm that the device has not been discovered. The MCTP bus owner then issues a

Set Endpoint ID command to the physical address for the endpoint which causes the endpoint to set its Discovered flag to *discovered*.

It is recommended that the MCTP bus owner maintains a list of the unique MCTP EIDs for the endpoints it has discovered and reassigns the same MCTP EIDs to those endpoints if an address changes during system operation.

Figure 10 provides an example flow of operations for partial endpoint discovery.

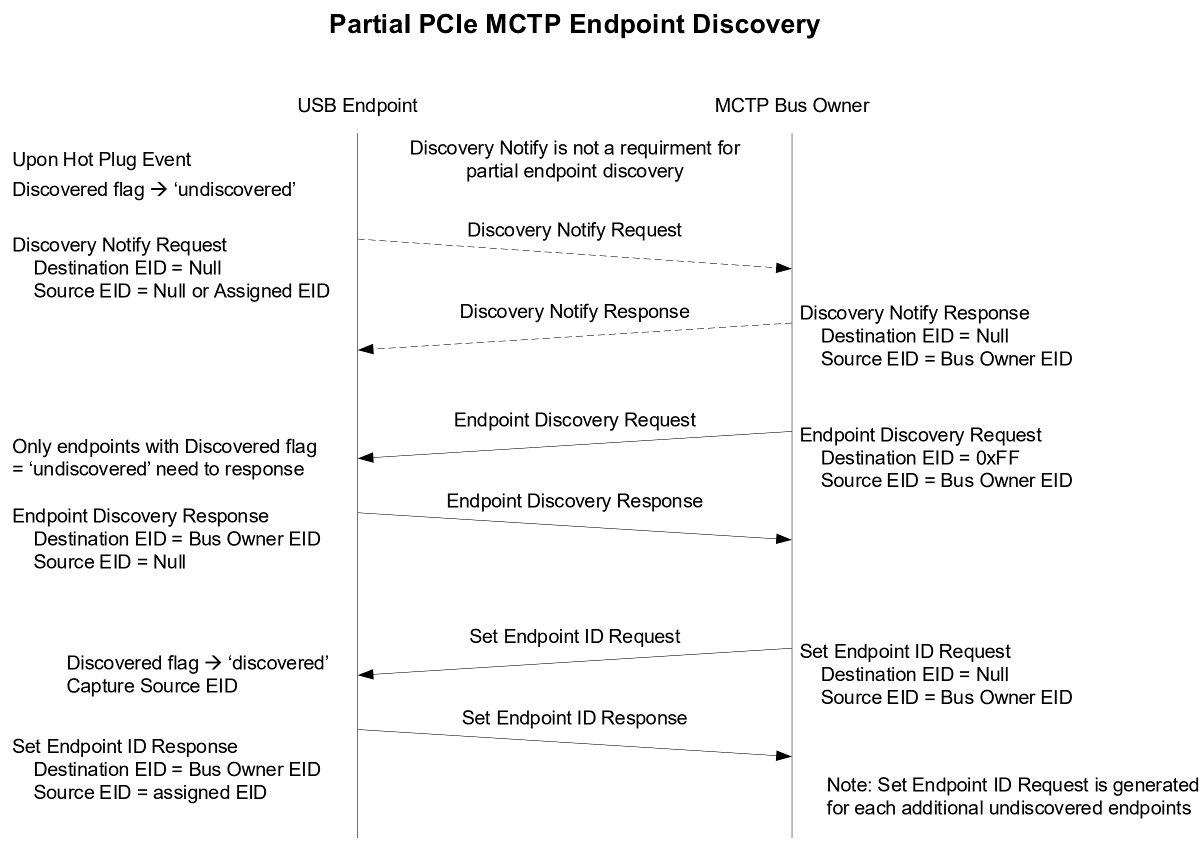

Figure 10 – Flow of Operations for Partial Endpoint Discovery

6.4.3 Endpoint Re-enumeration

If the bus implementation includes hot-plug devices, the bus owner shall perform a full or partial endpoint discovery any time the MCTP bus owner goes into a temporary state where the MCTP bus owner can miss receiving a Discovery Notify message (for example, if the bus owner device is reset or receives a firmware update). Whether a full or partial endpoint discovery is required is dependent on how much information the MCTP bus owner retains from prior enumerations.

6.5 Supported media

The transport binding defined in this specification has been designed to work with USB 2.0 compatible buses. The USB media type identifiers for this binding spec are defined in [Management Component Transport Protocol (MCTP) IDs and Codes](), in *MCTP physical medium identifiers* section.

6.6 MCTP Messages Routing and USB MCTP bridge

MCTP packet routing within a USB bus uses the USB root as an MCTP bridge for routing MCTP packets between MCTP endpoints.

6.7 Physical address of MCTP over USB packets

Per USB specifications, an MCTP over USB endpoint is addressed on the USB fabric using the combined 7-bits USB Device Address plus 4-bits Endpoint number. The Device Address is configured during the interface enumeration process as defined in USB Bus Enumeration chapter, while the endpoints numbers are defined in the endpoints descriptors as described in 6.1.4.2.1 and 6.1.4.2.2.

The Device Address and Endpoint number are only used in the Bulk transfer token as shown in Figure 6. As the MCTP over USB Header does not include the Device Address and does not include the Endpoint number, there is no need for any MCTP endpoint other than the MCTP over USB bus owner to record the endpoint address. The bus owner will always add the USB Device Address and Endpoint number of the destination endpoint to the USB Bulk packet that is sent to that endpoint.

Note: an MCTP over USB endpoint uses 2 Bulk endpoints with the same endpoint number, as described in section 6.1.4

The address format shown in Table 2 is used for MCTP control commands that require a physical address parameter to be returned for a bus that uses this transport binding. This includes commands such as the Resolve Endpoint ID, Routing Information Update, and Get Routing Table Entries commands.

Table 2 – Physical Address Format

Format Size	Layout and Description	
2 bytes	Byte 1	[7] – 0 [6:0] – USB Device Address
	Byte 2	[7:4] – 0000 [3:0] – Endpoint Number

6.8 Host dependencies

MCTP over USB is not dependent on the operational state of the host system and is operational in all power states S5 through S0. The USB bus is only reset on power on reset of the management controller or when USB Reset signaling is used as defined in USB.

6.9 Get endpoint ID medium-specific information

The medium-specific information as shown in Table 3 shall be used for the medium-specific Information field returned in the response to the Get Endpoint ID MCTP control message.

Table 3 – Medium-specific information

Description	
[7:0]	reserved

6.10 Composite devices

A composite device which integrates more than a single managed devices entities within the same physical device may assign a separate MCTP endpoint to each such managed device entity. In such a case, each MCTP endpoint shall use its own MCTP over USB endpoint interface using a shared USB connection.

6.11 MCTP over USB packet and control message timing requirements

In USB, all traffic passes through the USB Root.

Table 4 – Timing specifications for MCTP control messages on USB

Timing Specification	Symbol	Min	Max	Description
Endpoint ID reclaim	TRECLAIM	5 sec	-	Minimum time that a bus owner shall wait before reclaiming the EID for a non-responsive hot-plug endpoint (i.e., not ACKing repeated GETSTATUS CCCs).
Request-to-response time	MT1	–	100 ms	This interval is measured at the responder from the end of the reception of the MCTP Control Protocol request to the beginning of the transmission of the response. This requirement is tested under the condition where the responder can successfully transmit the response on the first try.
Time-out waiting for a response	MT2	MT1 max[1] + 2 * MT3 max	MT4, min[1]	This interval at the requester sets the minimum amount of time that a requester should wait before retrying a MCTP control request. This interval is measured at the requester from the end of the successful transmission of the MCTP control request to the beginning of the reception of the corresponding MCTP control response. NOTE: This specification does not preclude an implementation from adjusting the minimum time-out waiting for a response to a smaller number than MT2 based on the measured response times from responders. The mechanism for doing so is outside the scope of this specification.
Transmission Delay	MT3	-	100 ms	Allowed time between the end of the transmission of an MCTP Control Protocol message at the transmitter to the beginning of the reception of the MCTP Control Protocol message at the receiver.
Inter-Packet delay for Multi-Packet messages	MT3a	-	100 ms	Allowed time between the end of the transmission of an MCTP packet with EOM=0 to the beginning of the following MCTP packet of the same Message (see Message assembly in *Management Component Transport Protocol (MCTP) Base Specification*), measured at the transmitter

Timing Specification	Symbol	Min	Max	Description
Instance ID expiration interval	MT4	5 sec [2]	6 sec	Interval after which the instance ID for a given response will expire and become reusable if a response has not been received for the request. This is also the maximum time that a responder tracks an instance ID for a given request from a given requester.
NOTE 1: Unless otherwise specified, this timing applies to the mandatory and optional MCTP commands. NOTE 2: If a requester is reset, it may produce the same sequence number for a request as one that was previously issued. To guard against this, it is recommended that sequence number expiration be implemented. Any request from a given requester that is received more than MT4 seconds after a previous, matching request should be treated as a new request, not a retry.				

401

ANNEX A
(informative)

Change log

Version	Date	Description
1.0.0	2023-11-02	Initial release.
1.0.1	2024-05-21	Errata fixes Bulk Endpoint parameter • bEndpointAddress bit 7 shall be 1 for IN bulk endpoint per USB 2.0 spec. Typo correction in 6.4.1 Corrected document name