Document Number: DSP2037

Date: 2014-02-20

Version: 1.0.0

MCTP Packets and NC-SI over MCTP Overview

Document Type: White Paper

Document Status: DMTF Informational

Document Language: en-US

Copyright notice

Copyright © 2014 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents for uses consistent with this purpose, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit http://www.dmtf.org/about/policies/disclosures.php.

CONTENTS

Abstract 6
Foreword 7
  Acknowledgments 7
1 Introduction 9
  1.1 Target audience 9
  1.2 Related documents 9
  1.3 Terminology 10
  1.4 Acronyms and abbreviations 10
2 MCTP overview 11
  2.1 Bindings 11
  2.2 MCTP header 11
    2.2.1 Endpoint IDs 12
    2.2.2 Start of Message and End of Message bits 13
    2.2.3 Packet Sequence Number 13
    2.2.4 Tag Owner 13
    2.2.5 Message Tag 14
  2.3 Message assembly 14
  2.4 Interleaving of MCTP messages 14
  2.5 Message Payload and Message Payload header 15
    2.5.1 MCTP Vendor Defined Messages (VDM) 16
    2.5.2 Example MCTP and Message Payload headers 16
  2.6 Medium-specific header and trailer 18
    2.6.1 PCIe Vendor Defined Message 18
    2.6.2 SMBus/I2C 19
  2.7 Bus Owner 19
3 Sample flows and example packets 19
  3.1 Sample configuration 19
    3.1.1 SMBus setup 20
    3.1.2 PCIe setup 20
4 MCTP control commands 21
  4.1 Sample MCTP control messages 21
    4.1.1 MCTP Get Endpoint UUID command 21
    4.1.2 MCTP Get Endpoint UUID command over SMBus 21
    4.1.3 Get Endpoint UUID command over PCIe VDM 22
    4.1.4 MCTP Get Message Type Support 23
    4.1.5 MCTP Get Message Type Support command over SMBus 23
    4.1.6 MCTP Get Message Type Support command over PCIe VDM 24
5 NC-SI over MCTP 26
  5.1 NC-SI Commands over RBT packet formats 26
    5.1.1 NC-SI over RBT command packet 26
    5.1.2 NC-SI over RBT response packet 27
  5.2 Separation of NC-SI protocol and physical binding 27
  5.3 NC-SI over MCTP packets 27
    5.3.1 NC-SI command over MCTP 27
    5.3.2 NC-SI response over MCTP 28
    5.3.3 Comparing NC-SI Command packets over RBT and over MCTP 28
  5.4 NC-SI command over MCTP examples 29
    5.4.1 NC-SI Clear Initial State command over MCTP over SMBus 29
    5.4.2 NC-SI Clear Initial State command over MCTP over PCIe VDM 30
    5.4.3 NC-SI Set MAC Address command over MCTP over SMBus 30
    5.4.4 NC-SI Set MAC Address command over MCTP over PCIe VDM 32

6 NC-SI Pass-through (Ethernet) over MCTP 32
  6.1 Standard Ethernet/Pass-through packets 32
  6.2 NC-SI Pass-through/Ethernet over MCTP examples 34
    6.2.1 ARP request over MCTP over SMBus 35
    6.2.2 ARP request over MCTP over PCIe VDM 36
7 Example flows 36
  7.1 Discovery of NC-SI over MCTP capable endpoints 36
  7.2 Initialization of NC-SI over MCTP 38
  7.3 Transition from SMBus to PCIe VDM 38
  7.4 Transition from PCIe VDM to SMBus 40
8 Summary 41

Tables

Table 1 – MCTP header 12
Table 2 – MCTP header endpoint IDs 12
Table 3 – Special Endpoint IDs 12
Table 4 – MCTP flags 13
Table 5 – SOM and EOM combinations 13
Table 6 – MCTP message header 15
Table 7 – MCTP message types 15
Table 8 – MCTP header for an MCTP command request packet 16
Table 9 – MCTP header for an MCTP command response packet 16
Table 10 – Start Of Message example packet 17
Table 11 – Middle of message example packet 17
Table 12 – End Of Message example packet 17
Table 13 – PCIe VDM header 18
Table 14 – Sample MCTP over PCIe VDM packet header 18
Table 15 – PCIe VDM trailer 19
Table 16 – SMBus/I2C header 19
Table 17 – SMBus/I2C trailer 19
Table 18 – SMBus pins on PCIe connector 20
Table 19 – Get Endpoint UUID command over SMBus 21
Table 20 – Get Endpoint UUID response over SMBus 22
Table 21 – Get Endpoint UUID request over PCIe VDM 22
Table 22 – Get Endpoint UUID response over PCIe VDM 23
Table 23 – Get Message Type Support request over SMBus 24
Table 24 – Get Message Type Support response over SMBus 24
Table 25 – Get Message Type Support request over PCIe VDM 25
Table 26 – Get Message Type Support response over PCIe VDM 25
Table 27 – Clear Initial State command over MCTP over SMBus 29
Table 28 – Clear Initial State command over MCTP over PCIe VDM 30
Table 29 – Set MAC Address command over MCTP over SMBus 31
Table 30 – Set MAC Address command over MCTP over PCIe VDM 32
Table 31 – ARP request to MC over MCTP over SMBus 35
Table 32 – ARP request to MC over MCTP over PCIe VDM 36
Table 33 – Routing table entry format 37

Figures

Figure 1 – MCTP message encapsulation 11
Figure 2 – Sample SMBus setup for add-in card 20
Figure 3 – MCTP over PCIe VDM setup 20
Figure 4 – NC-SI over RBT command format 26
Figure 5 – NC-SI over RBT response format 27
Figure 6 – NC-SI command over MCTP packet 27
Figure 7 – NC-SI response over MCTP packet 28
Figure 8 – NC-SI command over RBT 28
Figure 9 – NC-SI command over MCTP 29
Figure 10 – Pass-through Ethernet packet format 33
Figure 11 – Ethernet packet over MCTP format 33
Figure 12 – Example MCTP topology 37
Figure 13 – Example Bus Owners 39

Abstract

The *Management Component Transport Protocol* (MCTP) (DSP0236) is a protocol defined by the DMTF Platform Management Component Intercommunications sub-team of the DMTF. MCTP is designed to support communications between different intelligent hardware components that make up a platform management subsystem that is provides monitoring and control functions inside a managed system. The *NC-SI over MCTP Binding Specification* (DSP0261) defines the binding of the NC-SI protocol over an MCTP physical binding.

This document provides an overview of MCTP over SMBus/I2C and MCTP over PCIe VDM packet formats and commands and provides descriptions and examples of how they are used with NC-SI over MCTP. The paper then presents information and examples of how NC-SI over MCTP may be used in a typical system.

Foreword

The *MCTP Packets and NC-SI over MCTP Overview* (DSP2037) 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 the DMTF, see http://www.dmtf.org.

Acknowledgments

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

Editor:

- Patrick Kutch – Intel Corporation

Contributors:

- Alan Berenbaum – Standard Microsystems
- Patrick Caporale – IBM Corporation
- Eliel Louzoun – Intel Corporation
- Yuval Itkin – Mellanox Technologies
- Patrick Schoeller – Hewlett-Packard Company
- Hemal Shah – Broadcom Corporation
- Tom Slaight – Intel Corporation

1 Introduction

The *MCTP Base Protocol Specification* (DSP0236) defines a mechanism by which a message payload can be sent independent of the physical medium. There are also a growing number of DMTF specifications detailing how to transmit packets over specific mediums, such as SMBus/I2C, PCI Express Vendor Defined Messaging, KCS, and Serial.

The MCTP specification itself defines a minimal set of commands used in configuring and reading the MCTP topology. While this is interesting, it in general does not provide anything of much use to a Management Controller (MC).

What makes MCTP so powerful and interesting is that the MCTP is designed to send any kind of payload. MCTP itself is agnostic to the data encapsulated within the packets; it places no restrictions or requirements on the payload within the messages.

The *NC-SI over MCTP Binding Specification* (DSP0261) provides a binding of the NC-SI protocol elements in order for NC-SI Control and Pass-through traffic to be transported over an MCTP connection.

This document will provide an overview of the MCTP packet format, as well as the packet format for MCTP packets over PCIe VDM and SMBus/I2C. Many sample packets from both MCTP over PCIe VDM and MCTP over SMBus/I2C will be discussed and compared. This document is not intended to be a replacement for the various MCTP specifications. Rather it should be used as a companion to the MCTP specifications.

Note that this document contains tables and descriptions from other MCTP specifications for the sake of simplicity and easy reference. These tables and descriptions are up to date at the time of the publication of this document. Please refer to the latest version of the actual specifications to ensure accuracy, because the information in formal specifications may change over time.

1.1 Target audience

The intended target audience for this document is readers who develop or utilize platform management subsystems that are formed by using management controllers and intelligent management devices and who are interested in obtaining an overview of the MCTP specifications that define a common intercommunication mechanism for those components.

This document is intended as a high-level introduction and overview of MCTP. It also provides numerous sample packets for MCTP commands, NC-SI over MCTP commands and Ethernet over MCTP packets over both SMBus and PCIe VDM bindings.

1.2 Related documents

DMTF DSP0222, *Network Controller Sideband Interface (NC-SI) Specification 1.0*
http://dmtf.org/sites/default/files/standards/documents/DSP0222_1.0.pdf

DMTF DSP0236, Management Component Transport Protocol (MCTP) Base Specification 1.2
http://www.dmtf.org/standards/published_documents/DSP0236_1.2.pdf

DMTF DSP0237, *Management Component Transport Protocol (MCTP) SMBus/I2C Transport Binding Specification 1.0*
http://www.dmtf.org/standards/published_documents/DSP0237_1.0.pdf

DMTF DSP0238, *Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding Specification 1.0*
http://dmtf.org/sites/default/files/standards/documents/DSP0238_1.0.pdf

DMTF DSP0239, Management Component Transport Protocol (MCTP) IDs and Codes 1.2
http://www.dmtf.org/standards/published_documents/DSP0239_1.2.pdf

DMTF DSP0261, *NC-SI over MCTP Binding Specification*
http://www.dmtf.org/standards/published_documents/DSP0261_1.0.pdf

SMBus, *System Management Bus (SMBus) Specification v2.0*, SMBus, 2000
http://www.smbus.org/specs/smbus20.pdf

PCI-SIG, *PCI Express Base Specification 3.0, PCIeV3.0,* November 10, 2010,
http://www.pcisig.com/specifications/pciexpress/base3/

1.3 Terminology

Term	**Definition**
I^{2}C	The name of a multi-master, two-wire, serial bus originally developed by Philips Semiconductor.
Management Controller	A microcontroller or processor that aggregates management parameters from one or more management devices and makes access to those parameters available to local or remote software, or to other management controllers, through one or more management data models. Management controllers may also interpret and process management-related data, and initiate management-related actions on management devices. While a native data model is defined for PMCI, it is designed to be capable of supporting other data models, such as CIM, IPMI, and vendor-specific data models. The microcontroller or processor that serves as a management controller can also incorporate the functions of a management device.
Managed Element	The finest granularity of addressing that can be the target of commands or messages, or a collection thereof.
Out-of-Band	Manageability capabilities that operate with hardware resources and components that are independent of operating systems control.
RMII	A reduced signal count MAC to PHY interface, based on the IEEE Media Independent Interface (MII), which was specified by the RMII Consortium (3Com Corporation; AMD Inc.; Bay Networks, Inc.; Broadcom Corp.; National Semiconductor Corp.; and Texas Instruments Inc.).
SMBus	The name of a multi-master, two-wire, serial bus specified by the Smart Battery Systems Implementer's Forum.

1.4 Acronyms and abbreviations

Term	**Definition**
MCTP	Management Component Transport Protocol
PCIe	PCI Express™
PMCI	Platform Management Component Intercommunications The name of the sub-team of the DMTF Pre-OS Working Group that developed the MCTP and other platform management hardware -related

Term	Definition
	specifications.
MC	Management Controller
RBT	RMII Based Transport
AEN	Asynchronous Event Notification

2 MCTP overview

MCTP provides a mechanism by which messages can be passed between two endpoints regardless of the physical connection (binding) between the endpoints. The content of the message is independent of the mechanism by which they are transmitted and received.

MCTP messages each contain the message payload, an MCTP header and a communication medium specific header, and possibly a trailer.

This clause provides an overview of many of the components of an MCTP message as well as the SMBus/I2C and PCIe VDM bindings. This is not intended to be a replacement for the actual specifications detailing these topics – please refer to the individual specifications for details.

2.1 Bindings

There are currently a number of physical binding specifications for MCTP, including but not limited to:

- SMBus/I2C
- PCIe VDM
- KCS
- Serial

This document will provide actual example MCTP messages from both SMBus/I2C and PCIe VDM bindings.

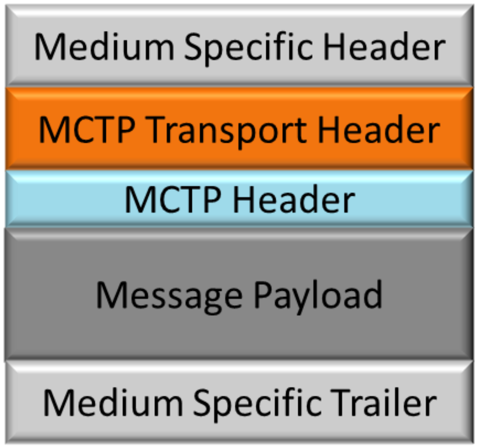


Figure 1 – MCTP message encapsulation

2.2 MCTP header

The MCTP header defines a routing and tracking mechanism for MCTP packets. This clause provides an overview of many of the components that make up the MCTP header. Please refer to the latest version of the specification for additional details.

Table 1 – MCTP header

<table>
<tr><td colspan="8">+0</td><td colspan="8">+1</td><td colspan="8">+2</td><td colspan="8">+3</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4" rowspan="2">RSVD</td><td colspan="4" rowspan="2">Header Version</td><td colspan="8" rowspan="2">Destination Endpoint ID</td><td colspan="8" rowspan="2">Source Endpoint ID</td><td colspan="8">MCTP Flags</td></tr>
<tr><td>SOM</td><td>EOM</td><td colspan="2">Packet seq #</td><td>TO</td><td colspan="3">Message Tag</td></tr>
</table>

The definitions of these fields are detailed in DSP0236.

2.2.1 Endpoint IDs

MCTP endpoint IDs are logical numbers assigned by the bus owner to an endpoint within an MCTP network. The numbers are unique within an MCTP network.

An endpoint may have different endpoint IDs assigned to it by the bus owners for different MCTP networks. A PCIe add-in card for example may have one endpoint ID for a SMBus/I2C MCTP network and a different endpoint ID for the PCIe VDM MCTP network.

Table 2 – MCTP header endpoint IDs

<table>
<tr><td colspan="8">+1</td><td colspan="8">+2</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Destination Endpoint ID</td><td colspan="8">Source Endpoint ID</td></tr>
</table>

2.2.1.1 Special endpoint IDs

The MCTP specification has reserved some EID values for specific purposes and for future use. Table 3 lists EID values that are reserved or assigned to specific functions for MCTP.

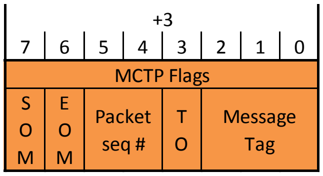
Table 3 – Special Endpoint IDs

Value	Description
Destination endpoint ID 0	**Null Destination EID**. This value indicates that the destination EID value is to be ignored and that only physical addressing is used to route the message to the destination on the given bus. This enables communication with devices that have not been assigned an EID. Because the physical addresses between buses are not guaranteed to be unique, MCTP does not support bridging messages with a null destination EID between different buses.
Source endpoint ID 0	**Null Source EID**. This value indicates a message is coming from an endpoint that is using physical addressing only. This would typically be used for messages that are delivered from an endpoint that has not been assigned an EID. Because the physical addresses between buses are not guaranteed to be unique, MCTP does not support bridging messages with a null source EID between different buses.
Endpoint IDs 1 through 7	Reserved for future definition.
Endpoint ID `0xFF`	**Broadcast EID**. Reserved for use as a broadcast EID on a given bus. MCTP network-wide broadcasts are not supported. Primarily for use by the MCTP Control message type.
All other values	Available for assignment and allocation to endpoints.

2.2.2 Start of Message and End of Message bits

Some messages are larger than the maximum transmit capability of the physical connection medium. To accommodate this, the MCTP header has fields to facilitate the breaking up of larger messages into multiple smaller packets.

Table 4 – MCTP flags



+3							
7	6	5	4	3	2	1	0
MCTP Flags							
SOM	EOM	Packet seq #		TO	Message Tag		

The Start Of Message (SOM) and End Of Message (EOM) bits are used to indicate whether a message is the start, end, or middle of a larger message that is being transmitted in smaller chunks. Table 5 provides a description of what the different combinations of the SOM and EOM bits indicate.

Table 5 – SOM and EOM combinations

SOM	EOM	Comment
1	1	Single packet message; Message Payload header field is present.
1	0	Beginning of multipacket message; Message Payload header field is present.
0	1	Last message of a multipacket message; Message Payload header field is not present.
0	0	Middle message of a multipacket message; Message Payload header field is not present.

All MCTP commands defined in the Base MCTP specification (DSP0236) fit within a single message. This means that each message will have both the Start of Message and End of Message bits set as well as the Message Payload header field.

The Message Payload header field is only present if the Start of Message bit is set. The destination endpoint will keep track of any packets that span multiple MCTP packets by using the Start of Message, End of Message, and Packet Sequence Number.

2.2.3 Packet Sequence Number

The Packet Sequence Number is used for messages that span multiple packets. The value for the Packet Sequence Number ranges from 0 to 3.

2.2.4 Tag Owner

The Tag Owner bit indicates that the Message Tag was created by the source of the message. The originator of this message sets this bit; if there is a response, the responder clears it.

Note that the base definition of Tag Owner has no connection to whether a given message is a request or a response. It only says whether a message has a Tag that the sender originated, or has a Tag that the receiver generated or specified.

In some cases, a given Message Type specification may overlay additional meaning and requirements on the use of the Message Tag, such as using it to indicate whether a message is a request, response, datagram, etc. As an example, MCTP Control messages overlay a relationship between the Message Tag usage and whether a given message is a request or a response.

2.2.5 Message Tag

This field, along with the Source Endpoint ID and the Tag Owner field, identifies a unique message at the MCTP transport level. The originator of a message should make sure that the Message Tag value is incremented for each new MCTP message.

Note that as with the Tag Owner, the definition of Message Tag has no connection with whether a given message is a request or a response.

2.3 Message assembly

Per the MCTP specification, the following fields and only the following fields are collectively used to identify the packets that belong to a given message for the purpose of message assembly on a particular destination endpoint.

- Message Tag
- Tag Owner
- Source Endpoint ID

Note that the Message Type is not one of the fields, and thus the Message Type field cannot be used to identify MCTP packets that belong to the same message. The reason for this is that if an MCTP message spans multiple packets, the Message Type field is only present in the first message.

However, using the Message Tag and Tag Owner fields provide a mechanism by which a device can interleave more than one MCTP message to an endpoint. By example, consider an endpoint that supports both MCTP Control messages and Ethernet; a single Ethernet message will generally span multiple MCTP packets. It may be desirable to be able to send an MCTP Control message while also sending a large Ethernet message.

In this example, the interleaving of these two messages from the same source to the same destination endpoint can be achieved by ensuring that the Message Tag field differs between the two messages and that the Tag Owner field is set for both.

Another example may be that there are two source endpoints both sending messages to the same destination endpoint over the same MCTP connection; both source endpoints may send a message with the same exact Message Tag. In this example, the destination endpoint would be able to differentiate the two messages by the Source Endpoint ID in the messages.

2.4 Interleaving of MCTP messages

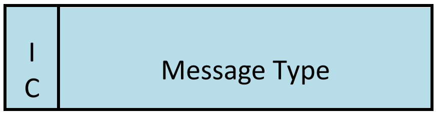
There are times when it may be desirable to interleave different MCTP messages. For example a Network Controller may be transmitting a Ethernet Pass-through Message spanning multiple MCTP packets to the Management Controller when a condition arises for which it should send an NC-SI AEN message to the Management Controller.

This is a legal operation; an endpoint may send an MCTP message of a different type interleaved within another message. It is not allowed by the specification, however, to interleave MCTP messages of the same type from the same source endpoint to the same destination endpoint. In that situation the source endpoint must finish transmitting the current MCTP message before sending the next message of the same message type.

2.5 Message Payload and Message Payload header

The type of data within the Message Payload is declared within the MCTP message header.

Table 6 – MCTP message header



The MCTP message header contains two fields. The Integrity Check (IC) bit (the MSB) indicates that there are integrity check bytes at the end of the message. The definition of the integrity check byte(s) is per the definition of the message type itself.

The Message Payload header only exists if the Start Of Message bit is set. For messages that span multiple MCTP packets, only the first packet will contain the Message Payload header.

There are currently four types of MCTP Payloads defined by the DMTF. The DSP0239 *Management Component Transport Protocol (MCTP) IDs and Codes* specification lists the following Payload Codes:

Table 7 – MCTP message types

Message Type	Message Type Code	Description
MCTP Control	`0x00`	Messages used to support initialization and configuration of MCTP communication within an MCTP network, as specified in DSP0236
Platform Level Data Model (PLDM)	`0x01`	Messages used to convey Platform Level Data Model (PLDM) traffic over MCTP.
NC-SI over MCTP	`0x02`	Messages used to convey NC-SI Control traffic over MCTP.
Ethernet over MCTP	`0x03`	Messages used to convey Ethernet traffic over MCTP.
Vendor Defined – PCI	`0x7E`	Message type used to support VDMs where the vendor is identifed by using a PCI-based vendor ID. The specification of the initial Message Header bytes for this message type is provided within this specification. The specification of the format of this message is given in DSP0236. Otherwise, the message body content is specified by the vendor, company, or organization identified by the given vendor ID.
Vendor Defined – IANA	`0x7F`	Message type used to support VDMs where the vendor is identifed by using an IANA-based vendor ID. This format uses an "Enterprise Number" that is assigned and maintained by the Internet Assigned Numbers Authority (IANA), www.iana.org, as the means of identifying a particular vendor, company, or organization. The specification of the format of this message is given in DSP0236. Otherwise, the message body content is specified by the vendor, company, or organization identified by the given vendor ID.
Reserved	all other	Reserved

The Base MCTP specification (DSP0236) itself defines the MCTP Control messages. These messages are generally for discovery and configuration of the MCTP Topology for endpoints.

2.5.1 MCTP Vendor Defined Messages (VDM)

The MCTP Vendor Defined Message types (0x7E and 0x7F) should not be confused with the PCIe Vendor Defined Messages (VDM) defined within the PCIe specification. They are both referred to as “Vendor Defined Messages”; however, they are very different.

MCTP messages over a PCIe physical binding are sent within PCIe VDMs using a specific PCIe VDM type. MCTP Vendor Defined Messages (VDMs) carry vendor-specific data that is encapsulated within the MCTP message itself and could be sent over any physical medium.

2.5.2 Example MCTP and Message Payload headers

This clause provides some example MCTP and Message Payload headers.

2.5.2.1 Single packet message

This clause shows the MCTP and Message headers for two single packet messages.

Table 8 – MCTP header for an MCTP command request packet

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4" rowspan="2">RSVD</td><td colspan="4" rowspan="2">Header Version
1</td><td colspan="8" rowspan="2">Destination Endpoint ID
0x12</td><td colspan="8" rowspan="2">Source Endpoint ID
0x32</td><td colspan="8">MCTP Flags</td></tr>
<tr><td>SOM
1</td><td>EOM
1</td><td colspan="2">Packet seq #
0</td><td>TO
1</td><td colspan="3">Message Tag
3</td></tr>
<tr><td>IC
0</td><td colspan="7">Message Type
0 (MCTP Control)</td></tr>
</table>

This message is transmitted over a single MCTP packet. It came from EID 0x12 and is being sent to EID 0x32. Both of the SOM (Start Of Message) and EOM (End Of Message) bits are set (indicating the complete message is contained within and it does not span multiple packets). The Tag Owner bit is also set, indicating that it was the source EID that determined the Message Tag. Note that the MCTP Message Type field is also present as the SOM bit is set. This particular example request (and subsequent response) is an MCTP Control messages.

Table 9 – MCTP header for an MCTP command response packet

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4" rowspan="2">RSVD</td><td colspan="4" rowspan="2">Header Version
1</td><td colspan="8" rowspan="2">Destination Endpoint ID
0x32</td><td colspan="8" rowspan="2">Source Endpoint ID
0x12</td><td colspan="8">MCTP Flags</td></tr>
<tr><td>SOM
1</td><td>EOM
1</td><td colspan="2">Packet seq #
0</td><td>TO
0</td><td colspan="3">Message Tag
3</td></tr>
<tr><td>IC
0</td><td colspan="7">Message Type
0 (MCTP Control)</td></tr>
</table>

Table 9 shows the MCTP header for the response to the request shown in Table 8. The destination and source EIDs are reversed from the request. The response message is also wholly contained within a single packet; as such both the SOM and EOM bits are set. The Message Tag in the response is the same as that of the request for tracking purposes. The Tag Owner bit is cleared, indicating that the source of this message was not the entity that set the value of the Message Tag.

Note that all MCTP Control messages, per the MCTP specification, will fit within a single packet.

2.5.2.2 Multipacket message

Table 10 – Start Of Message example packet

7 6 5 4	3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7	6	5 4	3	2 1 0
				MCTP Flags				
RSVD	Header Version 1	Destination Endpoint ID 0x32	Source Endpoint ID 0x12	SOM 1	EOM 0	Packet seq # 0	TO 1	Message Tag 6
IC 0	Message Type 3 (Ethernet)							

Table 10 is an MCTP Packet that is the beginning of a multipacket message. The SOM bit is set and the EOM bit is clear, indicating that it is the first Packet and more are to follow. The Message Type field is present (because the SOM bit is set) and indicates that this is an Ethernet message. The Packet Sequence Number is 0, and the Tag Owner bit is set to show that the source EID is the one that configured the Message Tag to be 6.

The destination EID uses the Message Type (in this case Ethernet) to handle the message after all the message packets have been received.

Table 11 – Middle of message example packet

7 6 5 4	3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7	6	5 4	3	2 1 0
				MCTP Flags				
RSVD	Header Version 1	Destination Endpoint ID 0x32	Source Endpoint ID 0x12	SOM 0	EOM 0	Packet seq # 1	TO 1	Message Tag 6

Table 11 is the second packet in this multipacket message example. Both the SOM and EOM bits are cleared, indicating that this packet is a middle packet, neither the start nor the end of the message. Because the SOM bit is cleared, the Message Type field is not present. The Tag Owner bit is still set because the source EID is the one that configured the Message Tag to be 6.

Note that the Message Tag value remains at 6 because the message is continuing; the Message Tag value should not be incremented until after a packet with the EOM bit is set. The Packet Sequence Number has been incremented by one to a value of 1.

Table 12 – End Of Message example packet

7 6 5 4	3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7	6	5 4	3	2 1 0
				MCTP Flags				
RSVD	Header Version 1	Destination Endpoint ID 0x32	Source Endpoint ID 0x12	SOM 0	EOM 1	Packet seq # 2	TO 1	Message Tag 6

Table 12 shows the MCTP header of the last packet in a multipacket message. The SOM bit is cleared while the EOM bit is set, indicating that this is the last packet for the message. The Packet Sequence Number has been incremented by one from the previous example. The Tag Owner and Message Tag fields remain the same as the previous packets because they are constant for the entire message that was split into three packets.

Because this packet is the last of the message, the destination endpoint must now combine the three packets into a single message for processing. Note that the Message Type field is only present in the first packet.

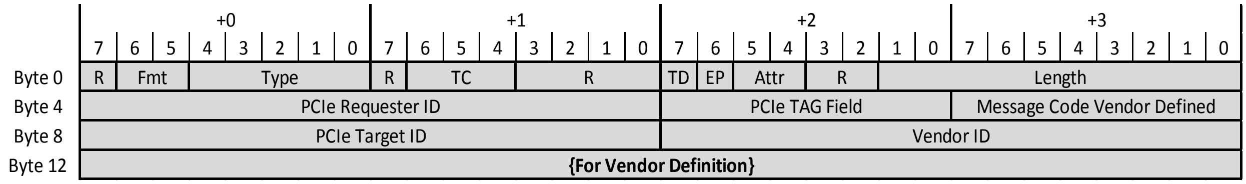
2.6 Medium-specific header and trailer

Different physical mediums, such as SMBus/I2C and PCIe (using VDM), have their own medium-specific header, and possibly trailer, requirements for data transmission. Each physical medium supported by MCTP has its own binding specification.

2.6.1 PCIe Vendor Defined Message

Table 13 shows the fields in a PCIe VDM packet according to the *PCI Express Base Specification*. Refer to the *PCI Express Base Specification* for details.

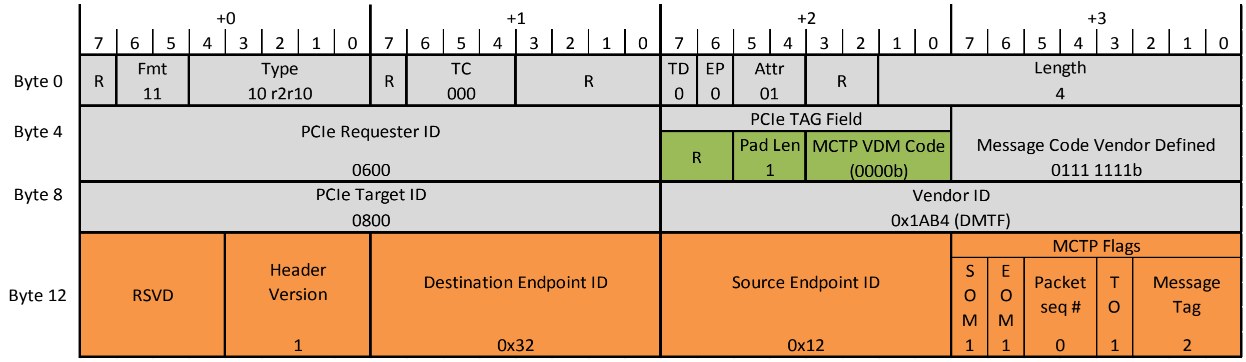
Table 13 – PCIe VDM header



	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	R	Fmt		Type					R	TC			R				TD	EP	Attr		R		Length									
Byte 4	PCIe Requester ID																PCIe TAG Field								Message Code Vendor Defined							
Byte 8	PCIe Target ID																Vendor ID															
Byte 12	**{For Vendor Definition}**																															

Table 14 provides an example of a PCIe VDM header for an MCTP packet. The *MCTP PCIe VDM Transport Binding Specification* (DSP0238) provides detailed information about each of the fields.

Table 14 – Sample MCTP over PCIe VDM packet header



	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	R	Fmt 11		Type 10 r2r10					R	TC 000			R				TD 0	EP 0	Attr 01		R		Length 4									
Byte 4	PCIe Requester ID 0600																PCIe TAG Field								Message Code Vendor Defined 0111 1111b							
																	R		Pad Len 1		MCTP VDM Code (0000b)											
Byte 8	PCIe Target ID 0800																Vendor ID 0x1AB4 (DMTF)															
Byte 12	RSVD				Header Version 1				Destination Endpoint ID 0x32								Source Endpoint ID 0x12								MCTP Flags							
																									SOM 1	EOM 1	Packet seq # 0		TO 1	Message Tag 2		

Comparing Table 13 and Table 14 one can see that the last four bytes of the PCIe VDM header in Table 13 are available for vendor definition. For MCTP over PCIe VDM messages, it was decided to use these last four bytes for the MCTP header as shown in Table 14.

The sample MCTP over PCIe VDM header shown in Table 14 indicates that there are four DWORDs of data that follow the PCIe VDM header. The source of the message is PCIe Requester ID 06:0:0, while the target is PCIe Target ID 08:0:0. There will be one pad byte at the end of the packet, which is the PCIe VDM trailer.

Note that the format (bits 6:5 of Byte 0) is 3, which indicates that the PCIe Header is four DWORDs long and that there is a payload following the header. Bytes 12 through 16 are part of the PCIe VDM header when the packet is of the type MCTP over PCIe VDM; these bytes are the MCTP header data.

Table 15 shows an example of a possible trailer for an MCTP packet over PCIe VDM. The PCIe specification requires that the payload be DWORD aligned, which means there may be from 0 to 3 padding bytes at the end of the payload for DWORD alignment.

Table 15 – PCIe VDM trailer

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">PCIe Padding
0x00</td></tr>
</table>

2.6.2 SMBus/I2C

Table 16 shows an example of a SMBus/I2C header for MCTP packets. The packet format conforms to the SMBus and I2C specifications. The command code is specific to MCTP, indicating that the packet is an MCTP packet.

Table 16 – SMBus/I2C header

<table>
<tr><th></th><th colspan="8">+0</th><th colspan="8">+1</th><th colspan="8">+2</th><th colspan="8">+3</th></tr>
<tr><th></th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>Byte 1</td><td colspan="8">Destination Slave Address</td><td colspan="8">Command Code
0x0F</td><td colspan="8">Byte Count</td><td colspan="8">Source Slave Address</td></tr>
</table>

Per the *MCTP SMBus/I2C Transport Binding Specification* (DSP0237), every MCTP over SMBus/I2C packet must contain the PEC (Packet Error Code) byte as defined in the *SMBus 2.0 Specification*. The PEC byte is the medium-specific trailer for SMBus/I2C.

Table 17 – SMBus/I2C trailer

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">PEC</td></tr>
</table>

2.7 Bus Owner

The Bus Owner has many responsibilities. Primary among them is to assign EIDs to the endpoints on the MCTP network(s). There can only be a single Bus Owner for each MCTP network; however, there can be multiple MCTP networks. For example, there can be an MCTP SMBus/I2C network as well as an MCTP PCIe VDM Fabric.

If an endpoint is part of two different MCTP networks, the endpoint ID does not need to be the same in both networks.

3 Sample flows and example packets

The previous clauses provided a high-level overview of MCTP along with the components of an MCTP message. Additionally the packet formats for both the MCTP over SMBus/I2C and MCTP over PCIe VDM were discussed.

The remaining clauses detail sample flows that the MC might perform. Each sample shows the MCTP packet over both SMBus and PCIe VDM.

3.1 Sample configuration

This clause describes the configuration used for the following examples. The MCTP endpoint is a NIC that supports both MCTP over SMBus/I2C and MCTP over PCIe VDM.

3.1.1 SMBus setup

The configuration for SMBus is shown in Figure 2.

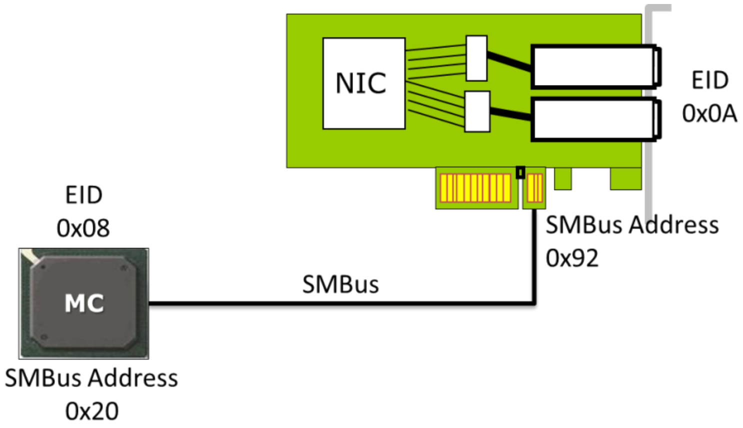


Figure 2 – Sample SMBus setup for add-in card

In this setup, the MC has the SMBus slave address of 0x20 and has been assigned the endpoint ID (EID) of 0x08 by the Bus Owner. The MC is connected to an add-in card (in this example an NIC) through the SMBus pins on the PCIe Connector, as shown in Table 18:

Table 18 – SMBus pins on PCIe connector

Pin	Side B Connector	
#	Name	Description
5	SMCLK	SMBus clock
6	SMDAT	SMBus data

Note that the SMBus/I2C MCTP connection is not limited to being over the PCIe connector.

3.1.2 PCIe setup

Figure 3 shows an example of an MCTP over PCIe VDM configuration.

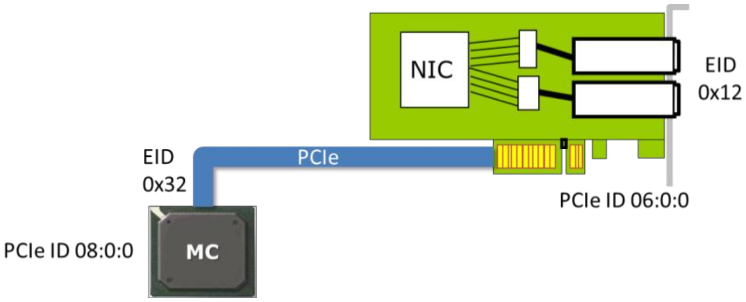


Figure 3 – MCTP over PCIe VDM setup

In this setup, the MC has a PCIe ID of 08:0:0 and was assigned an EID of 0x32 by the Bus Owner. The MC is connected to an endpoint (an NIC) with a PCIe ID of 06:0:0 and was assigned an EID of 0x12 by the Bus Owner.

4 MCTP control commands

The *Management Component Transport Protocol (MCTP) Base Specification* (DSP0236) defines a number of MCTP control commands. This clause details some sample MCTP command messages, showing what they look like in both MCTP over SMBus/I2C and MCTP over PCIe VDM.

4.1 Sample MCTP control messages

This clause provides some complete MCTP control message samples over SMBus/I2C and over PCIe VDM. Each sample command shows the raw, unformatted data followed by a table that decodes the messages.

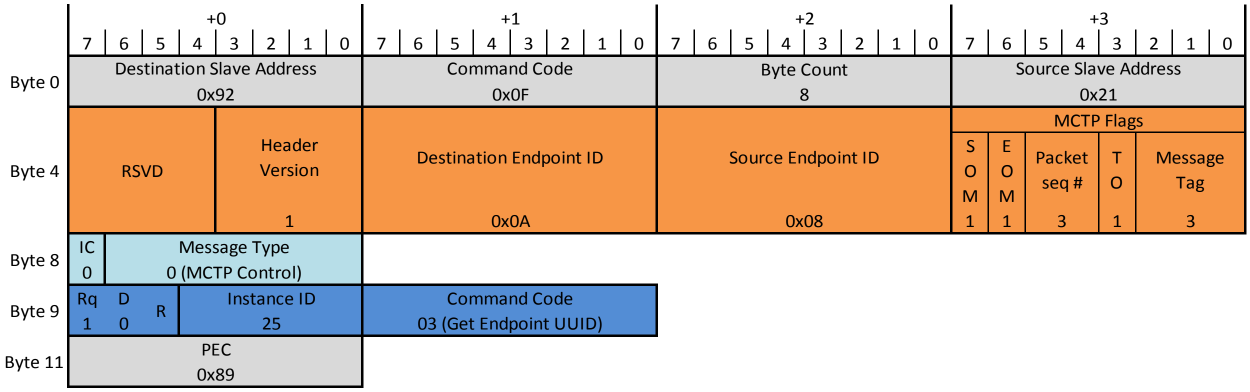
4.1.1 MCTP Get Endpoint UUID command

The MC may want to know the endpoint UUID (Universally Unique Identifier). To get this information, it issues the MCTP Get Endpoint UUID command. This clause provides examples what that request and response might look like over both SMBus and PCIe VDM.

4.1.2 MCTP Get Endpoint UUID command over SMBus

4.1.2.1 Request

Table 19 – Get Endpoint UUID command over SMBus



	+0	+1	+2	+3
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0	Destination Slave Address 0x92	Command Code 0x0F	Byte Count 8	Source Slave Address 0x21
Byte 4	RSVD Header Version 1	Destination Endpoint ID 0x0A	Source Endpoint ID 0x08	MCTP Flags SOM 1 EOM 1 Packet seq # 3 TO 1 Message Tag 3
Byte 8	IC 0 Message Type 0 (MCTP Control)			
Byte 9	Rq 1 D 0 R Instance ID 25	Command Code 03 (Get Endpoint UUID)		
Byte 11	PEC 0x89			

The payload for this MCTP message is the Get Endpoint UUID command request. We know it is a request because the request bit is set within the MCTP command header. The command code is 0x03, which indicates a Get Endpoint UUID command.

All MCTP control commands fit within a single packet, as such the Start of Message and End of Message bits are set.

4.1.2.2 Response

Table 20 – Get Endpoint UUID response over SMBus

	+0	+1	+2	+3
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0	Destination Slave Address 0x20	Command Code 0x0F	Byte Count 25	Source Slave Address 0x93
Byte 4	RSVD Header Version 1	Destination Endpoint ID 0x08	Source Endpoint ID 0x0A	MCTP Flags SOM 1 EOM 1 Packet seq # 0 TO 0 Message Tag 3
Byte 8	IC 0 Message Type 0 (MCTP Control)			
Byte 9	Rq 0 D 0 R Instance ID 25	Command Code 03 (Get Endpoint UUID)	Completion Code 00	Data 0x07
Byte 13	Data 0x75	Data 0xC0	Data 0x52	Data 0x19
Byte 17	Data 0xD4	Data 0xE0	Data 0x11	Data 0x4D
Byte 21	Data 0xBC	Data 0x00	Data 0xA0	Data 0xC9
Byte 25	Data 0x00	Data 0x00	Data 0x00	
Byte 28	PEC 0xC7			

Table 20 details the response from the Get Endpoint UUID command request over SMBus. Note that the Message Tag = 3 and the Tag Owner bit is 0, indicating that the Message Tag number did not originate from the source endpoint.

4.1.3 Get Endpoint UUID command over PCIe VDM

4.1.3.1 Request

Table 21 – Get Endpoint UUID request over PCIe VDM

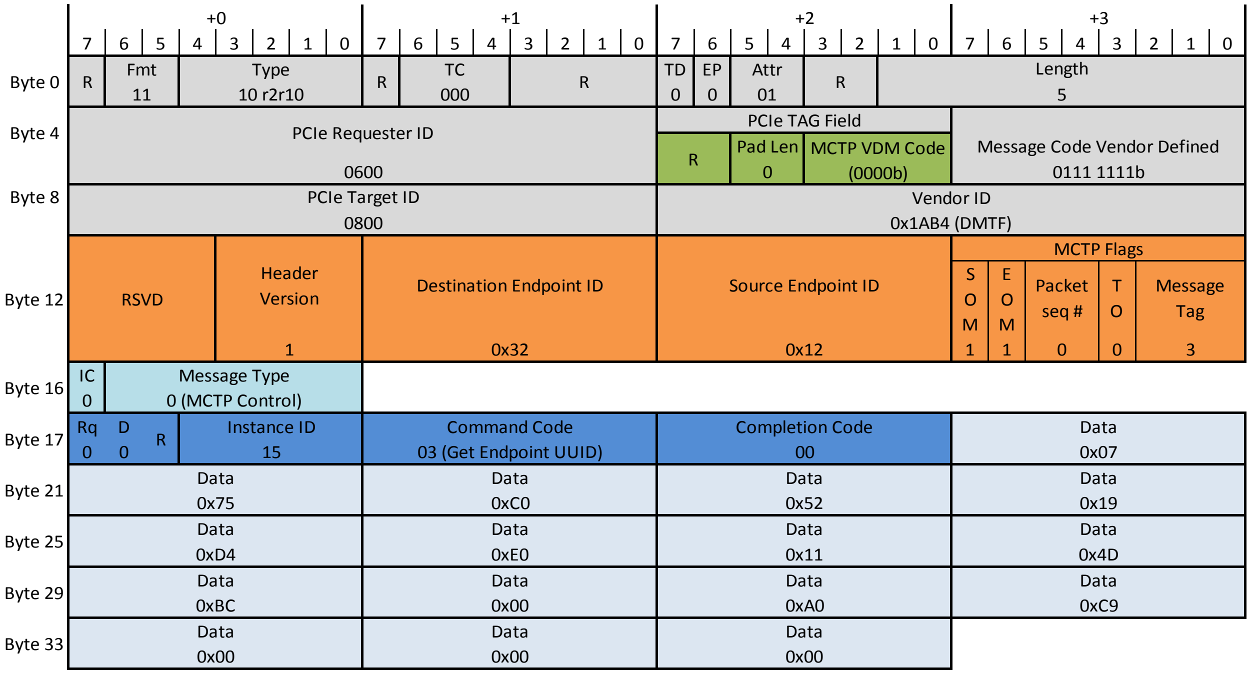
	+0	+1	+2	+3
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0	R Fmt 11 Type 10 r2r10	R TC 000 R	TD 0 EP 0 Attr 01 R Length 1	
Byte 4	PCIe Requester ID 0800		PCIe TAG Field R Pad Len 1 MCTP VDM Code (0000b)	Message Code Vendor Defined 0111 1111b
Byte 8	PCIe Target ID 0600		Vendor ID 0x1AB4 (DMTF)	
Byte 12	RSVD Header Version 1	Destination Endpoint ID 0x12	Source Endpoint ID 0x32	MCTP Flags SOM 1 EOM 1 Packet seq # 3 TO 1 Message Tag 3
Byte 16	IC 0 Message Type 0 (MCTP Control)			
Byte 17	Rq 1 D 0 R Instance ID 15	Command Code 03 (Get Endpoint UUID)		
Byte 19	PCIe Padding 0x00			

Table 21 details a sample MCTP Get Endpoint UUID command request. Note that many of the fields are the same as that from the MCTP Get Endpoint UUID command sent over SMBus, including:

- Start Of Message bit
- End Of Message Bit
- Tag Owner bit
- Message Type field
- MCTP command request bit
- MCTP command code

4.1.3.2 Response

Table 22 – Get Endpoint UUID response over PCIe VDM



	+0	+1	+2	+3
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0	R Fmt 11 Type 10 r2r10	R TC 000 R	TD 0 EP 0 Attr 01 R	Length 5
Byte 4	PCIe Requester ID 0600		PCIe TAG Field: R Pad Len 0 MCTP VDM Code (0000b)	Message Code Vendor Defined 0111 1111b
Byte 8	PCIe Target ID 0800		Vendor ID 0x1AB4 (DMTF)	
Byte 12	RSVD Header Version 1	Destination Endpoint ID 0x32	Source Endpoint ID 0x12	MCTP Flags: SOM 1 EOM 1 Packet seq # 0 TO 0 Message Tag 3
Byte 16	IC 0 Message Type 0 (MCTP Control)			
Byte 17	Rq 0 D 0 R Instance ID 15	Command Code 03 (Get Endpoint UUID)	Completion Code 00	Data 0x07
Byte 21	Data 0x75	Data 0xC0	Data 0x52	Data 0x19
Byte 25	Data 0xD4	Data 0xE0	Data 0x11	Data 0x4D
Byte 29	Data 0xBC	Data 0x00	Data 0xA0	Data 0xC9
Byte 33	Data 0x00	Data 0x00	Data 0x00	

The response to the MCTP Get Endpoint UUID over PCIe VDM is detailed in Table 22. Note that the actual MCTP response data is identical to the data received for the same response over SMBus in Table 20 with the exception of the instance ID.

4.1.4 MCTP Get Message Type Support

The MCTP Get Message Type Support command provides a mechanism by which the different types of MCTP messages that an endpoint supports can be determined. An MC might issue this command to determine whether an endpoint can support NC-SI over MCTP, for example.

4.1.5 MCTP Get Message Type Support command over SMBus

This clause provides MCTP over SMBus message examples, detailing both the request and response for the MCTP Get Message Type Support command.

4.1.5.1 Request

Table 23 – Get Message Type Support request over SMBus

	+0	+1	+2	+3
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0	Destination Slave Address 0x92	Command Code 0x0F	Byte Count 8	Source Slave Address 0x21
Byte 4	RSVD Header Version 1	Destination Endpoint ID 0x0A	Source Endpoint ID 0x08	MCTP Flags SOM 1 EOM 1 Packet seq # 3 TO 1 Message Tag 3
Byte 8	IC 0 Message Type 0 (MCTP Control)			
Byte 9	Rq 1 D 0 R Instance ID 26	Command Code 05 (Get Message Type Support)		
Byte 11	PEC 0xA4			

4.1.5.2 Response

Table 24 – Get Message Type Support response over SMBus

	+0	+1	+2	+3
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0	Destination Slave Address 0x92	Command Code 0x0F	Byte Count 12	Source Slave Address 0x93
Byte 4	RSVD Header Version 1	Destination Endpoint ID 0x08	Source Endpoint ID 0x0A	MCTP Flags SOM 1 EOM 1 Packet seq # 0 TO 0 Message Tag 3
Byte 8	IC 0 Message Type 0 (MCTP Control)			
Byte 9	Rq 0 D 0 R Instance ID 26	Command Code 05 (Get Message Type Support)	Completion Code 00	Data 0x02
Byte 13	Data 0x02	Data 0x03		
Byte 15	PEC 0x66			

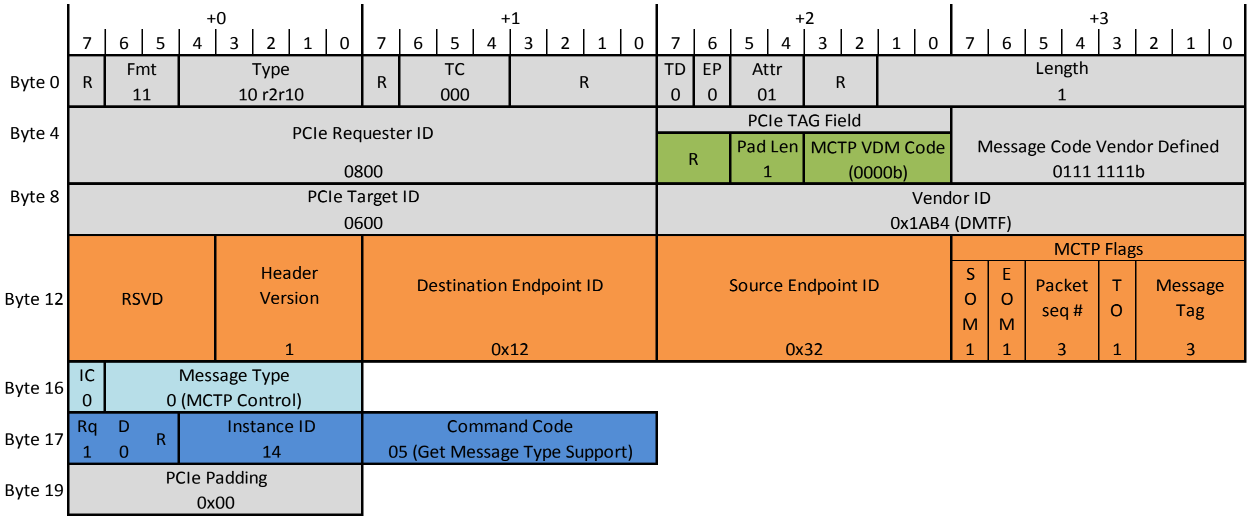
As Table 24 shows, the endpoint (a NIC) responded to the MCTP Get Message Type Support command request successfully. The returned data indicates that this endpoint supports two message types. Types 0x02 and 0x03 are indicated as being supported, which according to Table 7 correspond to NC-SI Command and NC-SI Pass-through (Ethernet) types.

4.1.6 MCTP Get Message Type Support command over PCIe VDM

This clause provides MCTP over PCIe VDM message examples, detailing both the request and response for the MCTP Get Message Type Support command.

4.1.6.1 Request

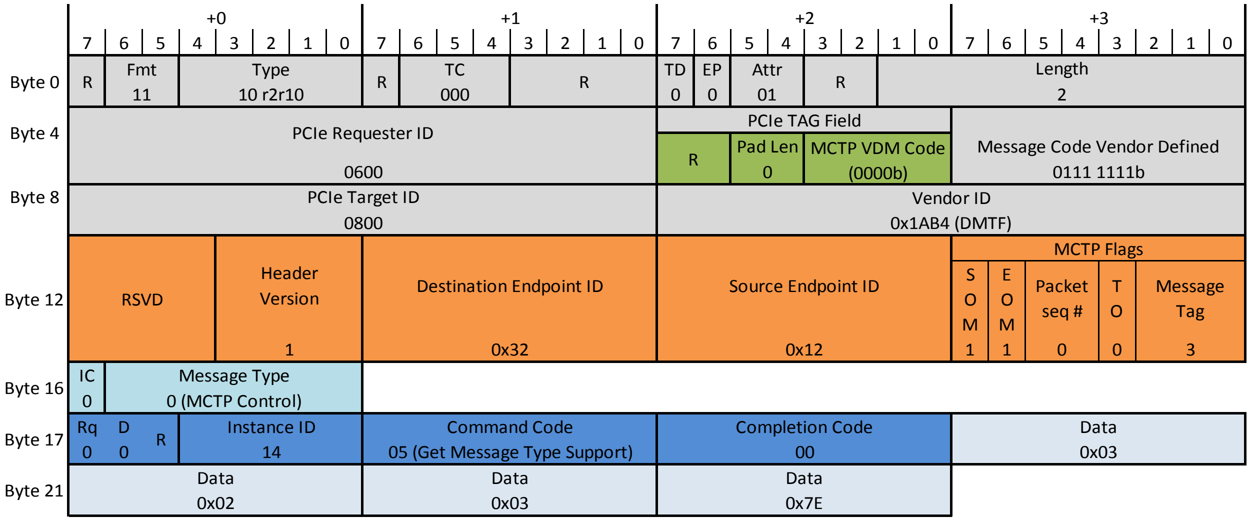
Table 25 – Get Message Type Support request over PCIe VDM



<table>
<tr><td></td><td colspan="8">+0</td><td colspan="8">+1</td><td colspan="8">+2</td><td colspan="8">+3</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Byte 0</td><td>R</td><td colspan="2">Fmt
11</td><td colspan="5">Type
10 r2r10</td><td>R</td><td colspan="3">TC
000</td><td colspan="4">R</td><td>TD
0</td><td>EP
0</td><td colspan="2">Attr
01</td><td colspan="2">R</td><td colspan="10">Length
1</td></tr>
<tr><td rowspan="2">Byte 4</td><td colspan="16" rowspan="2">PCIe Requester ID
0800</td><td colspan="8">PCIe TAG Field</td><td colspan="8" rowspan="2">Message Code Vendor Defined
0111 1111b</td></tr>
<tr><td colspan="2">R</td><td colspan="2">Pad Len
1</td><td colspan="4">MCTP VDM Code
(0000b)</td></tr>
<tr><td>Byte 8</td><td colspan="16">PCIe Target ID
0600</td><td colspan="16">Vendor ID
0x1AB4 (DMTF)</td></tr>
<tr><td rowspan="2">Byte 12</td><td colspan="4" rowspan="2">RSVD</td><td colspan="4" rowspan="2">Header Version
1</td><td colspan="8" rowspan="2">Destination Endpoint ID
0x12</td><td colspan="8" rowspan="2">Source Endpoint ID
0x32</td><td colspan="8">MCTP Flags</td></tr>
<tr><td>SOM
1</td><td>EOM
1</td><td colspan="2">Packet seq #
3</td><td>TO
1</td><td colspan="3">Message Tag
3</td></tr>
<tr><td>Byte 16</td><td>IC
0</td><td colspan="7">Message Type
0 (MCTP Control)</td></tr>
<tr><td>Byte 17</td><td>Rq
1</td><td>D
0</td><td>R</td><td colspan="5">Instance ID
14</td><td colspan="8">Command Code
05 (Get Message Type Support)</td></tr>
<tr><td>Byte 19</td><td colspan="8">PCIe Padding
0x00</td></tr>
</table>

4.1.6.2 Response

Table 26 – Get Message Type Support response over PCIe VDM



<table>
<tr><td></td><td colspan="8">+0</td><td colspan="8">+1</td><td colspan="8">+2</td><td colspan="8">+3</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Byte 0</td><td>R</td><td colspan="2">Fmt
11</td><td colspan="5">Type
10 r2r10</td><td>R</td><td colspan="3">TC
000</td><td colspan="4">R</td><td>TD
0</td><td>EP
0</td><td colspan="2">Attr
01</td><td colspan="2">R</td><td colspan="10">Length
2</td></tr>
<tr><td rowspan="2">Byte 4</td><td colspan="16" rowspan="2">PCIe Requester ID
0600</td><td colspan="8">PCIe TAG Field</td><td colspan="8" rowspan="2">Message Code Vendor Defined
0111 1111b</td></tr>
<tr><td colspan="2">R</td><td colspan="2">Pad Len
0</td><td colspan="4">MCTP VDM Code
(0000b)</td></tr>
<tr><td>Byte 8</td><td colspan="16">PCIe Target ID
0800</td><td colspan="16">Vendor ID
0x1AB4 (DMTF)</td></tr>
<tr><td rowspan="2">Byte 12</td><td colspan="4" rowspan="2">RSVD</td><td colspan="4" rowspan="2">Header Version
1</td><td colspan="8" rowspan="2">Destination Endpoint ID
0x32</td><td colspan="8" rowspan="2">Source Endpoint ID
0x12</td><td colspan="8">MCTP Flags</td></tr>
<tr><td>SOM
1</td><td>EOM
1</td><td colspan="2">Packet seq #
0</td><td>TO
0</td><td colspan="3">Message Tag
3</td></tr>
<tr><td>Byte 16</td><td>IC
0</td><td colspan="7">Message Type
0 (MCTP Control)</td></tr>
<tr><td>Byte 17</td><td>Rq
0</td><td>D
0</td><td>R</td><td colspan="5">Instance ID
14</td><td colspan="8">Command Code
05 (Get Message Type Support)</td><td colspan="8">Completion Code
00</td><td colspan="8">Data
0x03</td></tr>
<tr><td>Byte 21</td><td colspan="8">Data
0x02</td><td colspan="8">Data
0x03</td><td colspan="8">Data
0x7E</td></tr>
</table>

As Table 26 shows, the endpoint (a NIC) responded to the MCTP Get Message Type Support command request successfully. The returned data indicates that this endpoint supports three message types. Types 0x02, 0x03, and 0x7E are indicated as being supported, which according to Table 7 correspond to NC-SI Command, NC-SI Pass-through (Ethernet), and Vendor Defined - PCIe types.

Note that the response to this command differs between MCTP over SMBus/I2C and MCTP over PCIe VDM. Over SMBus, the Vendor Defined – PCIe message type is not supported; however, on this particular device, a Vendor Defined payload of some sort is supported.

The definition of what that Vendor Defined payload is beyond the scope of the DMTF specifications and of this document.

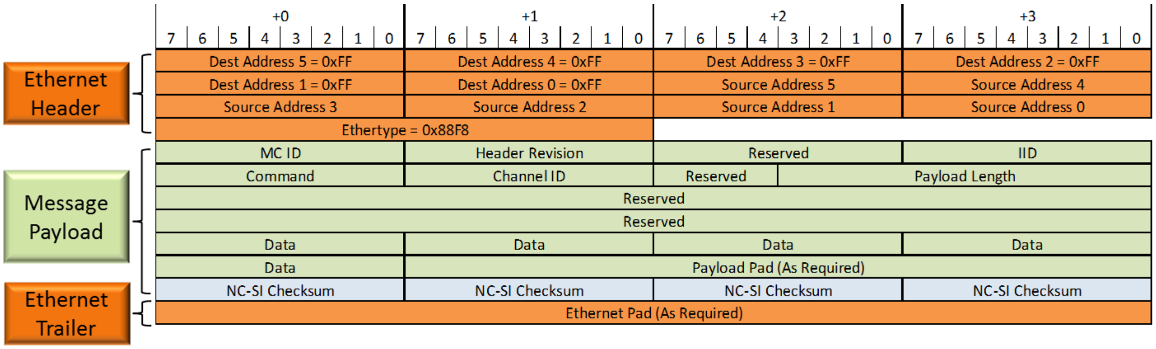
5 NC-SI over MCTP

The *NC-SI over MCTP Binding Specification* (DSP0261) details how to package NC-SI Commands over MCTP in addition to how to package NC-SI Pass-Through Ethernet traffic. This clause provides an overview of how NC-SI Commands over MCTP are packaged.

5.1 NC-SI Commands over RBT packet formats

Before delving into what NC-SI over MCTP looks like, let us first review what NC-SI over RBT packets look like.

5.1.1 NC-SI over RBT command packet



+0	+1	+2	+3
Dest Address 5 = 0xFF	Dest Address 4 = 0xFF	Dest Address 3 = 0xFF	Dest Address 2 = 0xFF
Dest Address 1 = 0xFF	Dest Address 0 = 0xFF	Source Address 5	Source Address 4
Source Address 3	Source Address 2	Source Address 1	Source Address 0
Ethertype = 0x88F8			
MC ID	Header Revision	Reserved	IID
Command	Channel ID	Reserved	Payload Length
Reserved			
Reserved			
Data	Data	Data	Data
Data	Payload Pad (As Required)		
NC-SI Checksum	NC-SI Checksum	NC-SI Checksum	NC-SI Checksum
Ethernet Pad (As Required)			

Figure 4 – NC-SI over RBT command format

Figure 4 shows the packet format of an NC-SI over RBT command. There are three basic components to the packet:

1) Ethernet header
2) NC-SI Command (Ethernet Message Payload)
3) Ethernet trailer

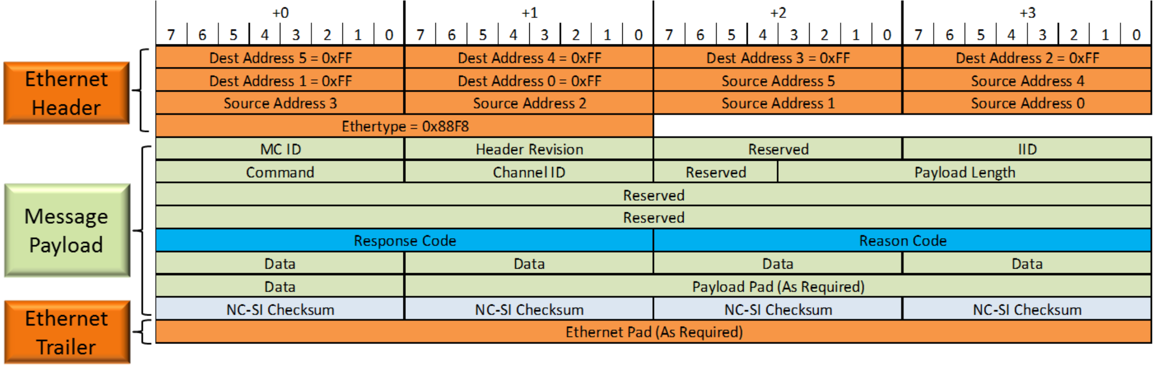
The Ethernet header comprises:

- Destination MAC address (all 0xFF's, per NC-SI specification)
- Source MAC address
- EtherType (0x88F8, per the NC-SI specification)

The NC-SI Command has a header of its own according to the NC-SI specification as well as optional data, padding and checksum.

The Ethernet trailer, if required, makes the entire Ethernet packet DWORD aligned.

5.1.2 NC-SI over RBT response packet



	+0	+1	+2	+3
Ethernet Header	Dest Address 5 = 0xFF	Dest Address 4 = 0xFF	Dest Address 3 = 0xFF	Dest Address 2 = 0xFF
	Dest Address 1 = 0xFF	Dest Address 0 = 0xFF	Source Address 5	Source Address 4
	Source Address 3	Source Address 2	Source Address 1	Source Address 0
	Ethertype = 0x88F8			
Message Payload	MC ID	Header Revision	Reserved	IID
	Command	Channel ID	Reserved	Payload Length
	Reserved			
	Reserved			
	Response Code		Reason Code	
	Data	Data	Data	Data
	Data	Payload Pad (As Required)		
	NC-SI Checksum	NC-SI Checksum	NC-SI Checksum	NC-SI Checksum
Ethernet Trailer	Ethernet Pad (As Required)			

Figure 5 – NC-SI over RBT response format

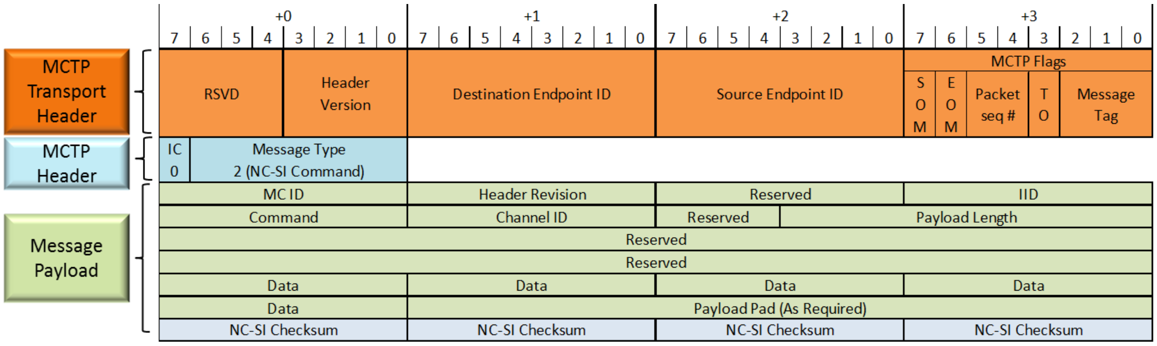
Figure 5 shows the packet format of a NC-SI over RBT response. It has all the same fields and requirements as the NC-SI over RBT command as discussed in 5.1.1. NC-SI responses contain two word-sized fields that are not present in the NC-SI Command: the response and reason codes.

5.2 Separation of NC-SI protocol and physical binding

The original NC-SI specification detailed both the physical (electrical) requirements as well as the protocol (command) definitions for NC-SI. Recognizing that the protocol portion of the NC-SI specification could be useful as a payload over an MCTP connection, the DMTF has since updated the NC-SI specification to include a separation for the NC-SI physical portion of the specification and the protocol portion. The terminology used now is NC-SI over the RMII Based Transport (RBT). This distinction between the physical connection requirements (RBT) and the NC-SI commands allows the use of NC-SI as an MCTP payload.

5.3 NC-SI over MCTP packets

5.3.1 NC-SI command over MCTP



	+0	+1	+2	+3
MCTP Transport Header	RSVD Header Version	Destination Endpoint ID	Source Endpoint ID	MCTP Flags: SOM EOM Packet seq # TO Message Tag
MCTP Header	IC 0 Message Type 2 (NC-SI Command)			
Message Payload	MC ID	Header Revision	Reserved	IID
	Command	Channel ID	Reserved	Payload Length
	Reserved			
	Reserved			
	Data	Data	Data	Data
	Data	Payload Pad (As Required)		
	NC-SI Checksum	NC-SI Checksum	NC-SI Checksum	NC-SI Checksum

Figure 6 – NC-SI command over MCTP packet

Figure 6 shows the format of an NC-SI command over MCTP. It consists of:

- MCTP transport header (see 2.2)
- MCTP header (see 2.5)
- MCTP command (the payload)

5.3.2 NC-SI response over MCTP

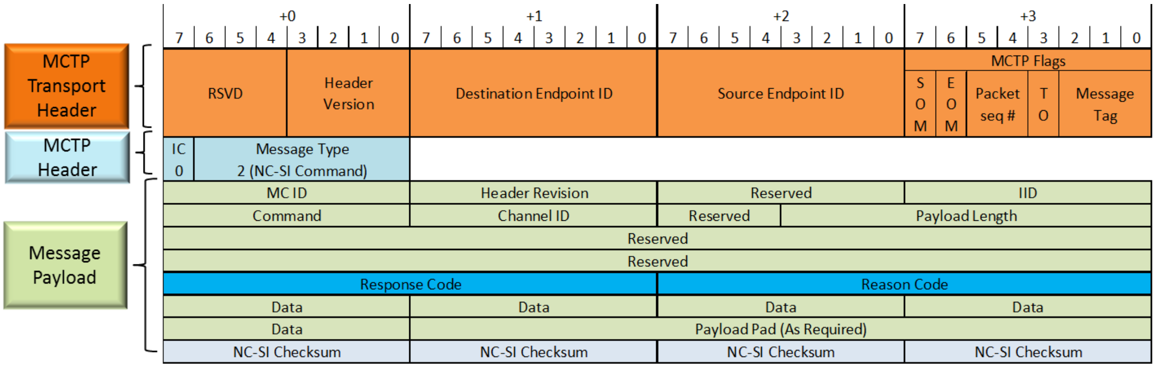


Figure 7 – NC-SI response over MCTP packet

As you can see in Figure 7, a NC-SI Response of MCTP has the same components as an NC-SI Command over MCTP:

- MCTP transport header (see 2.2)
- MCTP header (see 2.5)
- MCTP command response (the payload)

In the case of the response, the response and reason codes are present within the NC-SI response.

5.3.3 Comparing NC-SI Command packets over RBT and over MCTP

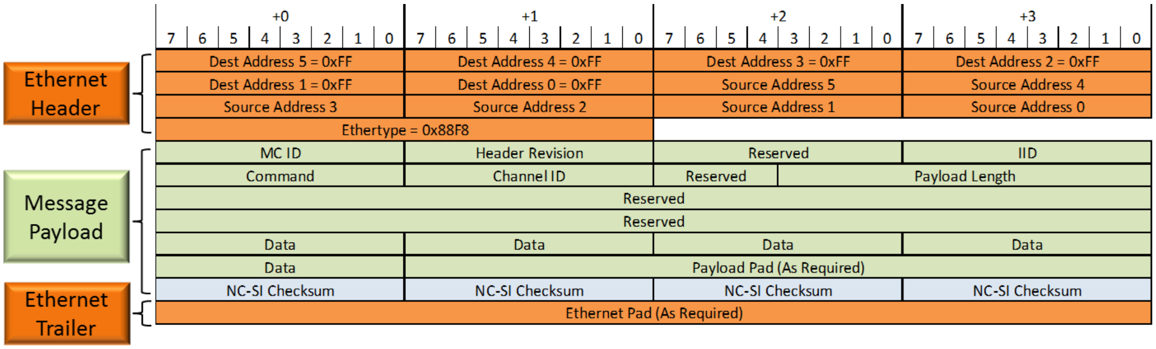
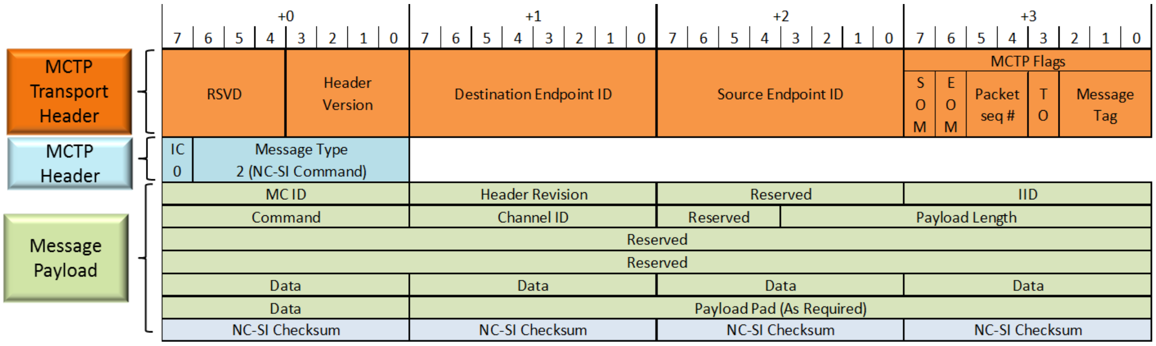


Figure 8 – NC-SI command over RBT



	+0 7 6 5 4 3 2 1 0	+1 7 6 5 4 3 2 1 0	+2 7 6 5 4 3 2 1 0	+3 7 6 5 4 3 2 1 0
MCTP Transport Header	RSVD Header Version	Destination Endpoint ID	Source Endpoint ID	MCTP Flags SOM EOM Packet seq # TO Message Tag
MCTP Header	IC 0 Message Type 2 (NC-SI Command)			
Message Payload	MC ID	Header Revision	Reserved	IID
	Command	Channel ID	Reserved Payload Length	
	Reserved			
	Reserved			
	Data	Data	Data	Data
	Data	Payload Pad (As Required)		
	NC-SI Checksum	NC-SI Checksum	NC-SI Checksum	NC-SI Checksum

Figure 9 – NC-SI command over MCTP

Figure 8 shows the packet format for a NC-SI command over RMII Based Transport. Figure 9 shows the packet format for a NC-SI command over MCTP. Note that the payload is the same; however, the transport headers differ.

The NC-Si command over RBT has the Ethernet header and trailer, while the NC-SI command over MCTP has the MCTP transport header and the MCTP header indicating the payload is an NC-SI command. The MCTP transport medium-specific header and trailer are not shown for simplicity sake.

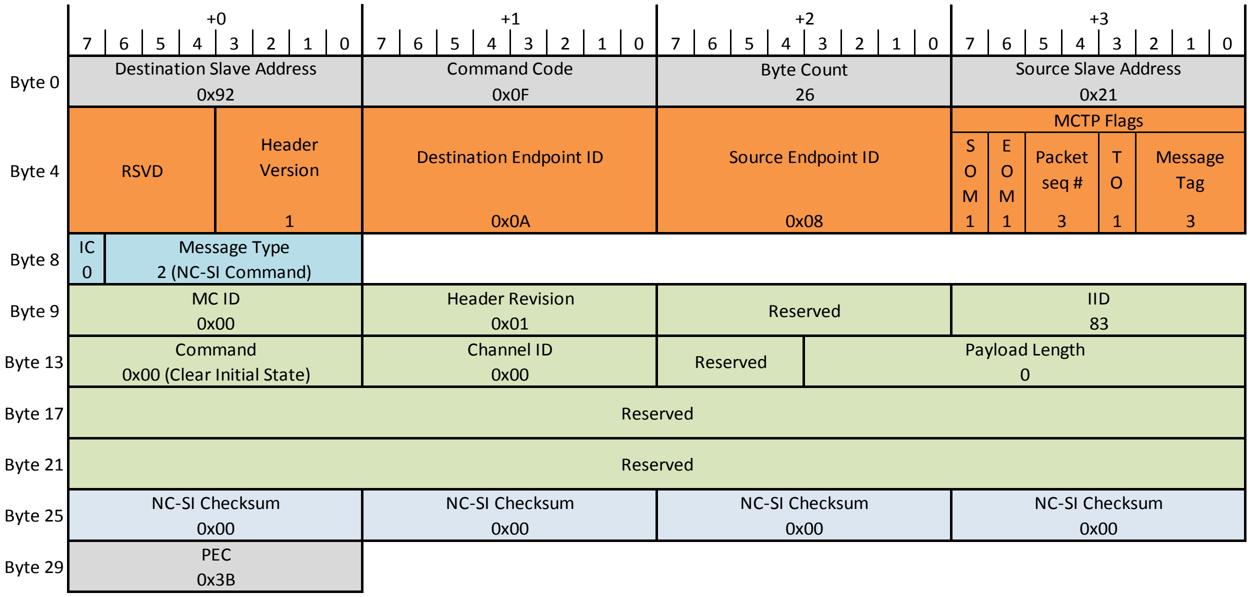
5.4 NC-SI command over MCTP examples

5.4.1 NC-SI Clear Initial State command over MCTP over SMBus

The first example will be the simplest of the NC-SI commands – Clear Initial State, which has no data associated with the command. The Clear Initial State command is exactly as is defined in the NC-SI specification (DSP0261).

5.4.1.1 Request

Table 27 – Clear Initial State command over MCTP over SMBus



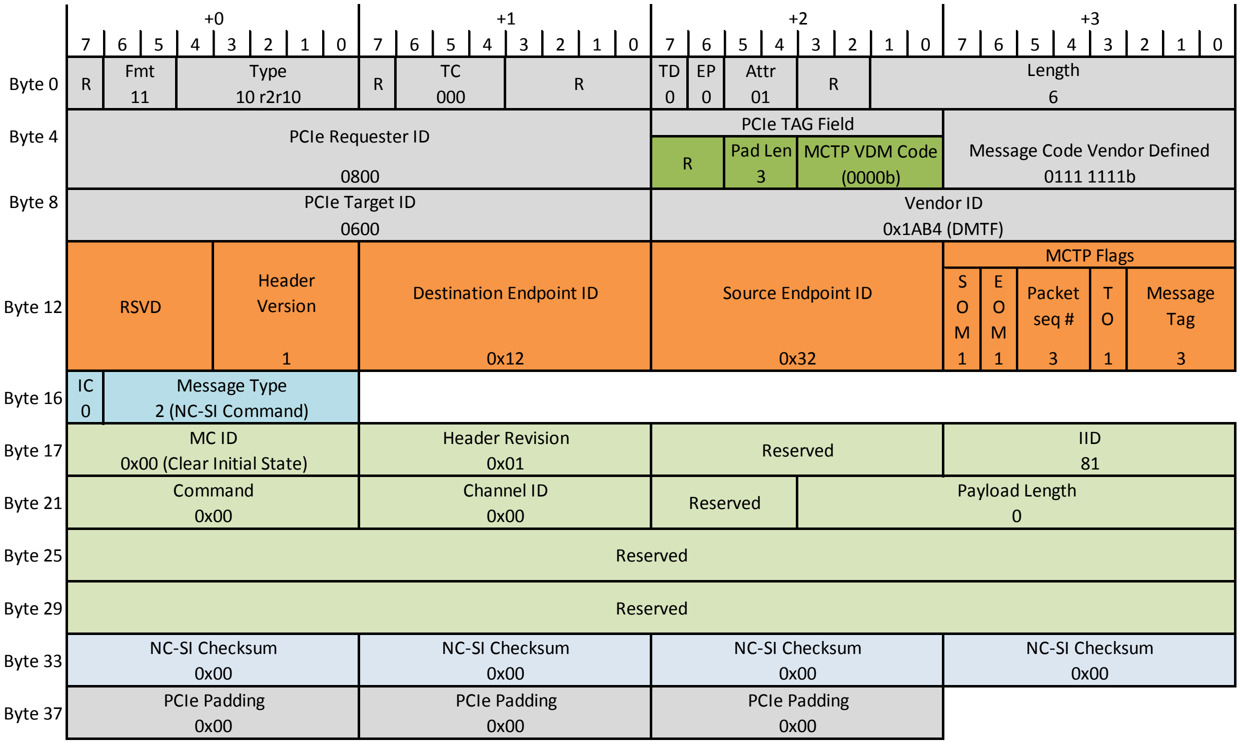
	+0 7 6 5 4 3 2 1 0	+1 7 6 5 4 3 2 1 0	+2 7 6 5 4 3 2 1 0	+3 7 6 5 4 3 2 1 0
Byte 0	Destination Slave Address 0x92	Command Code 0x0F	Byte Count 26	Source Slave Address 0x21
Byte 4	RSVD Header Version 1	Destination Endpoint ID 0x0A	Source Endpoint ID 0x08	MCTP Flags SOM 1 EOM 1 Packet seq # 3 TO 1 Message Tag 3
Byte 8	IC 0 Message Type 2 (NC-SI Command)			
Byte 9	MC ID 0x00	Header Revision 0x01	Reserved	IID 83
Byte 13	Command 0x00 (Clear Initial State)	Channel ID 0x00	Reserved Payload Length 0	
Byte 17	Reserved			
Byte 21	Reserved			
Byte 25	NC-SI Checksum 0x00	NC-SI Checksum 0x00	NC-SI Checksum 0x00	NC-SI Checksum 0x00
Byte 29	PEC 0x3B			

Just as in the MCTP Control examples, note how the Start of Message and End Of Message bits are set, indicating that this message is contained within a single packet. Also note that the NC-SI checksum fields are present within the NC-SI command. This is because those fields are part of the NC-SI message itself (which remains wholly intact in NC-SI over MCTP).

5.4.2 NC-SI Clear Initial State command over MCTP over PCIe VDM

5.4.2.1 Request

Table 28 – Clear Initial State command over MCTP over PCIe VDM



<table>
<tr><td></td><td colspan="8">+0</td><td colspan="8">+1</td><td colspan="8">+2</td><td colspan="8">+3</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Byte 0</td><td>R</td><td colspan="2">Fmt
11</td><td colspan="5">Type
10 r2r10</td><td>R</td><td colspan="3">TC
000</td><td colspan="4">R</td><td>TD
0</td><td>EP
0</td><td colspan="2">Attr
01</td><td colspan="2">R</td><td colspan="10">Length
6</td></tr>
<tr><td rowspan="2">Byte 4</td><td colspan="16" rowspan="2">PCIe Requester ID
0800</td><td colspan="8">PCIe TAG Field</td><td colspan="8" rowspan="2">Message Code Vendor Defined
0111 1111b</td></tr>
<tr><td colspan="2">R</td><td colspan="2">Pad Len
3</td><td colspan="4">MCTP VDM Code
(0000b)</td></tr>
<tr><td>Byte 8</td><td colspan="16">PCIe Target ID
0600</td><td colspan="16">Vendor ID
0x1AB4 (DMTF)</td></tr>
<tr><td rowspan="2">Byte 12</td><td colspan="4" rowspan="2">RSVD</td><td colspan="4" rowspan="2">Header Version
1</td><td colspan="8" rowspan="2">Destination Endpoint ID
0x12</td><td colspan="8" rowspan="2">Source Endpoint ID
0x32</td><td colspan="8">MCTP Flags</td></tr>
<tr><td>SOM
1</td><td>EOM
1</td><td colspan="2">Packet seq #
3</td><td>TO
1</td><td colspan="3">Message Tag
3</td></tr>
<tr><td>Byte 16</td><td>IC
0</td><td colspan="7">Message Type
2 (NC-SI Command)</td><td colspan="24"></td></tr>
<tr><td>Byte 17</td><td colspan="8">MC ID
0x00 (Clear Initial State)</td><td colspan="8">Header Revision
0x01</td><td colspan="8">Reserved</td><td colspan="8">IID
81</td></tr>
<tr><td>Byte 21</td><td colspan="8">Command
0x00</td><td colspan="8">Channel ID
0x00</td><td colspan="4">Reserved</td><td colspan="12">Payload Length
0</td></tr>
<tr><td>Byte 25</td><td colspan="32">Reserved</td></tr>
<tr><td>Byte 29</td><td colspan="32">Reserved</td></tr>
<tr><td>Byte 33</td><td colspan="8">NC-SI Checksum
0x00</td><td colspan="8">NC-SI Checksum
0x00</td><td colspan="8">NC-SI Checksum
0x00</td><td colspan="8">NC-SI Checksum
0x00</td></tr>
<tr><td>Byte 37</td><td colspan="8">PCIe Padding
0x00</td><td colspan="8">PCIe Padding
0x00</td><td colspan="8">PCIe Padding
0x00</td><td colspan="8"></td></tr>
</table>

Although the transport and MCTP headers differ from the same command being sent over SMBus (see 5.4.1.1), the actual NC-SI Command in the MCTP Payload is, with the exception of the IID, exactly the same. Remember that MCTP provides a mechanism to send a payload over different mediums.

5.4.3 NC-SI Set MAC Address command over MCTP over SMBus

This next example is the Set MAC Address command. The example sets and enables the first MAC address for Package 0, Channel 0 to be 00-25-90-7e-91-e5.

5.4.3.1 Request

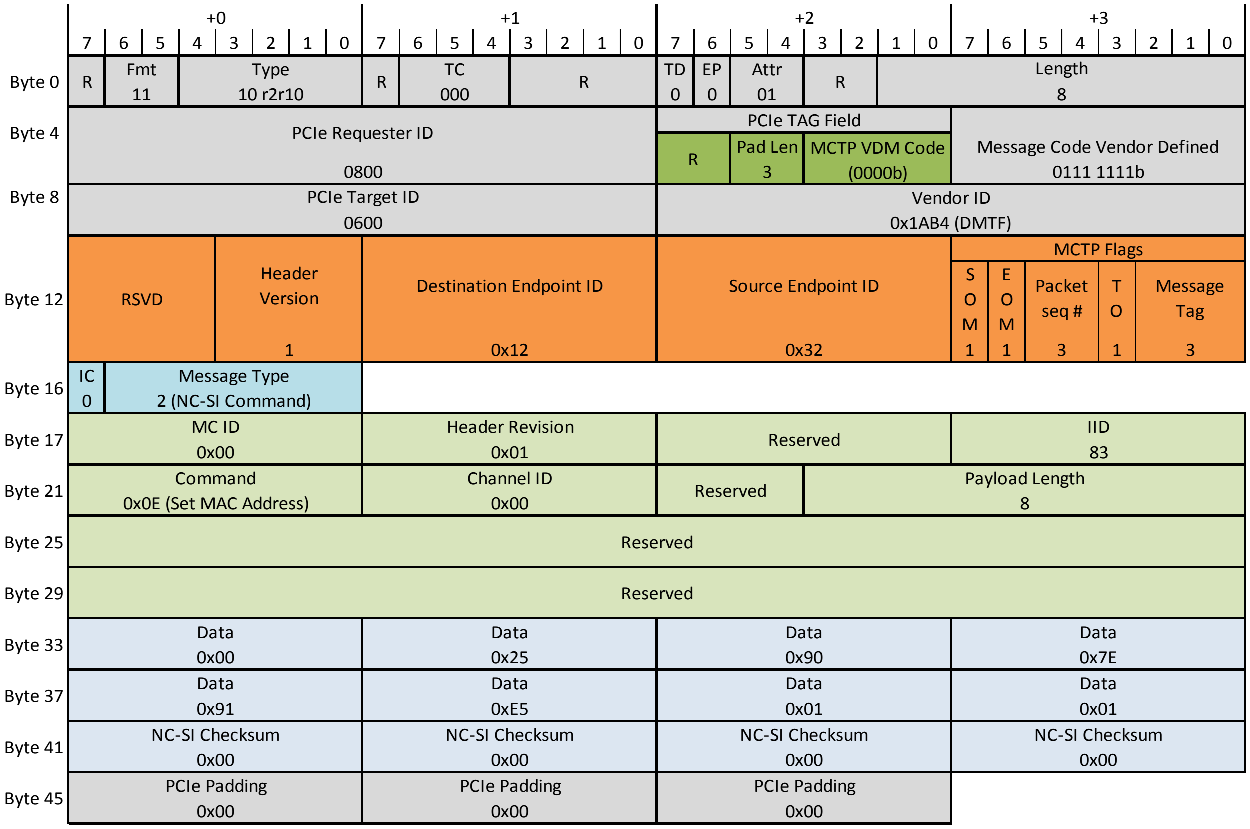
Table 29 – Set MAC Address command over MCTP over SMBus

<table>
<tr><th></th><th colspan="8">+0</th><th colspan="8">+1</th><th colspan="8">+2</th><th colspan="8">+3</th></tr>
<tr><th></th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>Byte 0</td><td colspan="8">Destination Slave Address
0x92</td><td colspan="8">Command Code
0x0F</td><td colspan="8">Byte Count
34</td><td colspan="8">Source Slave Address
0x21</td></tr>
<tr><td rowspan="2">Byte 4</td><td colspan="4" rowspan="2">RSVD</td><td colspan="4" rowspan="2">Header Version
1</td><td colspan="8" rowspan="2">Destination Endpoint ID
0x0A</td><td colspan="8" rowspan="2">Source Endpoint ID
0x08</td><td colspan="8">MCTP Flags</td></tr>
<tr><td>SOM
1</td><td>EOM
1</td><td colspan="2">Packet seq #
3</td><td>TO
1</td><td colspan="3">Message Tag
3</td></tr>
<tr><td>Byte 8</td><td>IC
0</td><td colspan="7">Message Type
2 (NC-SI Command)</td><td colspan="24"></td></tr>
<tr><td>Byte 9</td><td colspan="8">MC ID
0x00</td><td colspan="8">Header Revision
0x01</td><td colspan="8">Reserved</td><td colspan="8">IID
86</td></tr>
<tr><td>Byte 13</td><td colspan="8">Command
0x0E (Set MAC Address)</td><td colspan="8">Channel ID
0x00</td><td colspan="4">Reserved</td><td colspan="12">Payload Length
8</td></tr>
<tr><td>Byte 17</td><td colspan="32">Reserved</td></tr>
<tr><td>Byte 21</td><td colspan="32">Reserved</td></tr>
<tr><td>Byte 25</td><td colspan="8">Data
0x00</td><td colspan="8">Data
0x25</td><td colspan="8">Data
0x90</td><td colspan="8">Data
0x7E</td></tr>
<tr><td>Byte 29</td><td colspan="8">Data
0x91</td><td colspan="8">Data
0xE5</td><td colspan="8">Data
0x01</td><td colspan="8">Data
0x01</td></tr>
<tr><td>Byte 33</td><td colspan="8">NC-SI Checksum
0x00</td><td colspan="8">NC-SI Checksum
0x00</td><td colspan="8">NC-SI Checksum
0x00</td><td colspan="8">NC-SI Checksum
0x00</td></tr>
<tr><td>Byte 37</td><td colspan="8">PEC
0x1A</td><td colspan="24"></td></tr>
</table>

5.4.4 NC-SI Set MAC Address command over MCTP over PCIe VDM

5.4.4.1 Request

Table 30 – Set MAC Address command over MCTP over PCIe VDM



<table>
<tr><td></td><td colspan="8">+0</td><td colspan="8">+1</td><td colspan="8">+2</td><td colspan="8">+3</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Byte 0</td><td>R</td><td colspan="2">Fmt
11</td><td colspan="5">Type
10 r2r10</td><td>R</td><td colspan="3">TC
000</td><td colspan="4">R</td><td>TD
0</td><td>EP
0</td><td colspan="2">Attr
01</td><td colspan="2">R</td><td colspan="10">Length
8</td></tr>
<tr><td rowspan="2">Byte 4</td><td colspan="16" rowspan="2">PCIe Requester ID
0800</td><td colspan="8">PCIe TAG Field</td><td colspan="8" rowspan="2">Message Code Vendor Defined
0111 1111b</td></tr>
<tr><td colspan="2">R</td><td colspan="2">Pad Len
3</td><td colspan="4">MCTP VDM Code
(0000b)</td></tr>
<tr><td>Byte 8</td><td colspan="16">PCIe Target ID
0600</td><td colspan="16">Vendor ID
0x1AB4 (DMTF)</td></tr>
<tr><td rowspan="2">Byte 12</td><td colspan="4" rowspan="2">RSVD</td><td colspan="4" rowspan="2">Header Version
1</td><td colspan="8" rowspan="2">Destination Endpoint ID
0x12</td><td colspan="8" rowspan="2">Source Endpoint ID
0x32</td><td colspan="8">MCTP Flags</td></tr>
<tr><td>SOM
1</td><td>EOM
1</td><td colspan="2">Packet seq #
3</td><td>TO
1</td><td colspan="3">Message Tag
3</td></tr>
<tr><td>Byte 16</td><td>IC
0</td><td colspan="7">Message Type
2 (NC-SI Command)</td><td colspan="24"></td></tr>
<tr><td>Byte 17</td><td colspan="8">MC ID
0x00</td><td colspan="8">Header Revision
0x01</td><td colspan="8">Reserved</td><td colspan="8">IID
83</td></tr>
<tr><td>Byte 21</td><td colspan="8">Command
0x0E (Set MAC Address)</td><td colspan="8">Channel ID
0x00</td><td colspan="4">Reserved</td><td colspan="12">Payload Length
8</td></tr>
<tr><td>Byte 25</td><td colspan="32">Reserved</td></tr>
<tr><td>Byte 29</td><td colspan="32">Reserved</td></tr>
<tr><td>Byte 33</td><td colspan="8">Data
0x00</td><td colspan="8">Data
0x25</td><td colspan="8">Data
0x90</td><td colspan="8">Data
0x7E</td></tr>
<tr><td>Byte 37</td><td colspan="8">Data
0x91</td><td colspan="8">Data
0xE5</td><td colspan="8">Data
0x01</td><td colspan="8">Data
0x01</td></tr>
<tr><td>Byte 41</td><td colspan="8">NC-SI Checksum
0x00</td><td colspan="8">NC-SI Checksum
0x00</td><td colspan="8">NC-SI Checksum
0x00</td><td colspan="8">NC-SI Checksum
0x00</td></tr>
<tr><td>Byte 45</td><td colspan="8">PCIe Padding
0x00</td><td colspan="8">PCIe Padding
0x00</td><td colspan="8">PCIe Padding
0x00</td><td colspan="8"></td></tr>
</table>

As with the Clear Initial State example, except for IID and potentially the NC-SI checksum, the actual payload of the NC-SI over MCTP command does not differ from the MCTP over SMBus and MCTP over PCIe VDM.

6 NC-SI Pass-through (Ethernet) over MCTP

Clause 5.3 provides information about NC-SI Commands and how the actual NC-SI Command data is the same, whether it is packaged and sent over RBT or MCTP. This clause provides information about the second type of NC-SI traffic – Pass-through packets, which are Ethernet packets going to and from the MC.

6.1 Standard Ethernet/Pass-through packets

NC-SI was designed so that the NC-SI Commands sent between the MC and the NC are valid Ethernet packets. The commands are of a different EtherType (0x88F8) to distinguish them from other kinds of traffic. Ethernet traffic to and from the MC over NC-SI are just that, standard Ethernet packets.

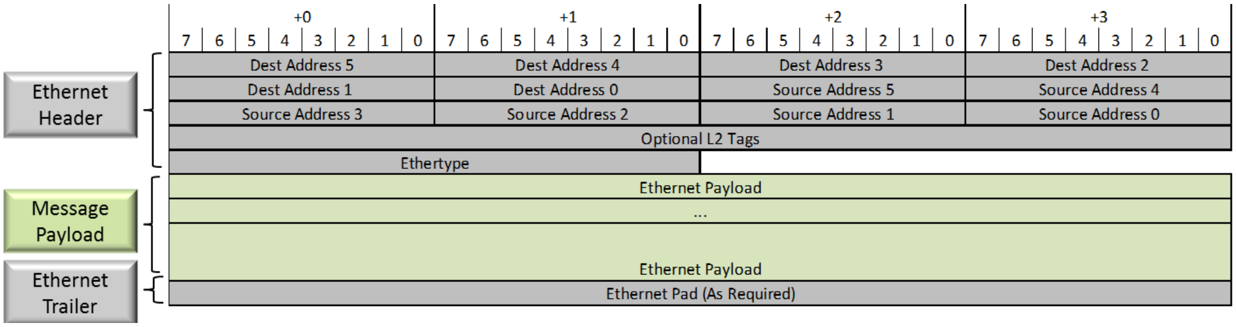


Figure 10 – Pass-through Ethernet packet format

Figure 10 shows the format of a standard Ethernet frame. It consists of:

- Destination address
- Source address
- Optional L2 tags
- EtherType
- Payload
- Ethernet pad

The destination and source addresses are 6-byte MAC addresses. The Optional Layer 2 tags are for VLAN and other such L2 tags. The EtherType indicates what kind of data are within the payload and the Ethernet trailer pads the frame to at least 64 bytes in length and DWORD alignment.

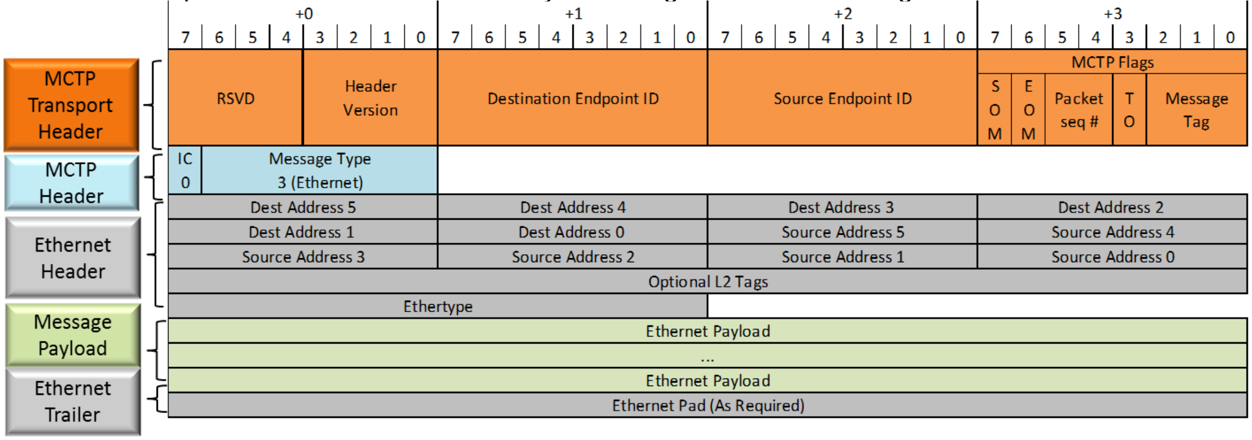


Figure 11 – Ethernet packet over MCTP format

Figure 11 details an Ethernet packet (NC-SI Pass-through) over MCTP. As with all MCTP packets, it contains the MCTP header. According to Table 7, the MCTP header indicates that the payload is an Ethernet frame. The actual Ethernet data format is identical to a standard Ethernet frame, as shown in 6.1, containing:

- Destination address
- Source address

- Optional L2 tags
- EtherType
- Payload
- Ethernet pad

If the Ethernet message is larger than the physical transport maximum size, the message is transmitted in multiple packets using the mechanism discussed in 2.5.2.2. Note that the MCTP Header field is only present if the Start of Message bit is set.

6.2 NC-SI Pass-through/Ethernet over MCTP examples

This clause details an example of an ARP request that has been received by the NC and is being transmitted to the MC. Both the example via MCTP over SMBus and MCTP over PCIe VDM fit within a single message.

6.2.1 ARP request over MCTP over SMBus

Table 31 – ARP request to MC over MCTP over SMBus

	+0	+1	+2	+3
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0	Destination Slave Address 0x20	Command Code 0x0F	Byte Count 66	Source Slave Address 0x93
Byte 4	RSVD Header Version 1	Destination Endpoint ID 0x08	Source Endpoint ID 0x0A	MCTP Flags SOM 1 EOM 1 Packet seq # 0 TO 1 Message Tag 4
Byte 8	IC 0 Message Type 3 (Ethernet)			
Byte 9	Data 0xFF	Data 0xFF	Data 0xFF	Data 0xFF
Byte 13	Data 0xFF	Data 0xFF	Data 0xA0	Data 0x36
Byte 17	Data 0x9F	Data 0x17	Data 0xF0	Data 0xF6
Byte 21	Data 0x08	Data 0x06	Data 0x00	Data 0x01
Byte 25	Data 0x08	Data 0x00	Data 0x06	Data 0x04
Byte 29	Data 0x00	Data 0x01	Data 0xA0	Data 0x36
Byte 33	Data 0x9F	Data 0x17	Data 0xF0	Data 0xF6
Byte 37	Data 0xC0	Data 0xA8	Data 0x20	Data 0x20
Byte 41	Data 0x00	Data 0x00	Data 0x00	Data 0x00
Byte 45	Data 0x00	Data 0x00	Data 0xC0	Data 0xA8
Byte 49	Data 0x20	Data 0x02	Data 0x00	Data 0x00
Byte 53	Data 0x00	Data 0x00	Data 0x00	Data 0x00
Byte 57	Data 0x00	Data 0x00	Data 0x00	Data 0x00
Byte 61	Data 0x00	Data 0x00	Data 0x00	Data 0x00
Byte 65	Data 0x00	Data 0x00	Data 0x00	Data 0x00
Byte 69	PEC 0xFF			

6.2.2 ARP request over MCTP over PCIe VDM

Table 32 – ARP request to MC over MCTP over PCIe VDM

	+0	+1	+2	+3
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0	R Fmt 11 Type 10 r2r10	R TC 000 R	TD 0 EP 0 Attr 01 R	Length 16
Byte 4	PCIe Requester ID 0600		PCIe TAG Field: R Pad Len 3 MCTP VDM Code (0000b)	Message Code Vendor Defined 0111 1111b
Byte 8	PCIe Target ID 0800		Vendor ID 0x1AB4 (DMTF)	
Byte 12	RSVD Header Version 1	Destination Endpoint ID 0x32	Source Endpoint ID 0x12	MCTP Flags: SOM 1 EOM 1 Packet seq # 0 TO 1 Message Tag 2
Byte 16	IC 0 Message Type 3 (Ethernet)			
Byte 17	Data 0xFF	Data 0xFF	Data 0xFF	Data 0xFF
Byte 21	Data 0xFF	Data 0xFF	Data 0x00	Data 0x1B
Byte 25	Data 0x21	Data 0x37	Data 0x40	Data 0x44
Byte 29	Data 0x08	Data 0x06	Data 0x00	Data 0x01
Byte 33	Data 0x08	Data 0x00	Data 0x06	Data 0x04
Byte 37	Data 0x00	Data 0x01	Data 0x00	Data 0x1B
Byte 41	Data 0x21	Data 0x37	Data 0x40	Data 0x44
Byte 45	Data 0xC0	Data 0xA8	Data 0x20	Data 0x02
Byte 49	Data 0x00	Data 0x00	Data 0x00	Data 0x00
Byte 53	Data 0x00	Data 0x00	Data 0xC0	Data 0xA8
Byte 57	Data 0x20	Data 0x20	Data 0x00	Data 0x00
Byte 61	Data 0x00	Data 0x00	Data 0x00	Data 0x00
Byte 65	Data 0x00	Data 0x00	Data 0x00	Data 0x00
Byte 69	Data 0x00	Data 0x00	Data 0x00	Data 0x00
Byte 73	Data 0x00	Data 0x00	Data 0x00	Data 0x00
Byte 77	PCIe Padding 0x00	PCIe Padding 0x00	PCIe Padding 0x00	

7 Example flows

This clause provides a number of example actions that the MC might take.

7.1 Discovery of NC-SI over MCTP capable endpoints

This example demonstrates a mechanism by which the MC may discover an MCTP endpoint that supports the NC-SI over MCTP binding. This is a generic example; the physical binding does not matter.

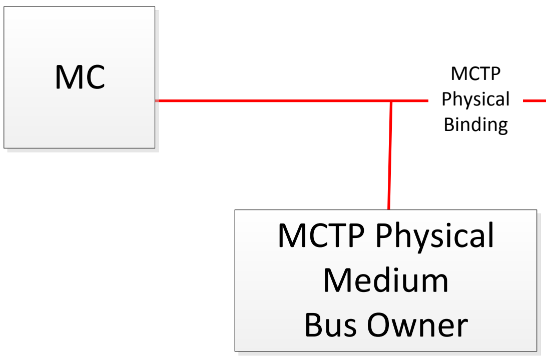


Figure 12 – Example MCTP topology

The MC has been enumerated (been assigned an EID) by the Bus Owner over a given medium. This medium will be used throughout the rest of the example for all MCTP control messages.

Determining what kinds of messages an endpoint supports is a straightforward process in which the MC issues the MCTP Get Message Type Support command. However before the MC can issue this command, it must first know the EID and physical address of the endpoint in order to issue the command.

Step 1 – MC issues the Get Routing Table Entries MCTP command to the Bus Owner. This will return a table of entries described in Table 33.

Table 33 – Routing table entry format

Byte	Description
1	Size of EID range associated with this entry
2	Starting EID
3	Entry Type/Port Number [7:6] – Entry Type: `00b` = Entry corresponds to a single endpoint that does not operate as an MCTP bridge. `01b` = Entry reflects an EID range for a bridge where the starting EID is the EID of the bridge itself and additional EIDs in the range are routed by the bridge. `10b` = Entry is for a single endpoint that serves as an MCTP bridge. `11b` = Entry is an EID range for a bridge, but does not include the EID of the bridge itself. [5] – Dynamic/Static Entry Indicates whether the entry was dynamically created or statically configured. Note that statically configured routing information shall not be merged with dynamic information when entry information is reported by using this command. While an implementation may internally organize its data that way, dynamic and statically configured routing must be reported as separate entries. Dynamically created entries include entries that were generated from the Routing Information Update command as well as entries that were created as a result of the bridge doing EID assignment and EID pool allocation as a Bus Owner. `0b` = Entry was dynamically created. `1b` = Entry was statically configured. [4:0] – Port Number This value is chosen by the bridge device vendor and is used to identify a particular bus connection under which the physical address for the entry is defined. In some cases, this number may correspond to an internal "logical" bus that is not directly connected to an external physical bus. Port numbers are required to be static. It is recommended, but not required, that the ports (bus connections) on the bridge be numbered sequentially starting from `0x00`. This specification does not define any requirements

Byte	Description
	or recommendations on how port numbers are assigned to corresponding physical connections on a device.
4	Physical Transport Binding Type Identifier, according to DSP0239
5	Physical Media Type Identifier, according to DSP0239 This value is used to indicate in what format the following physical address data is given.
6	Physical Address Size The size in bytes of the following Physical Address field The size is defined as part of the corresponding physical transport binding specification identified by the physical media type identifier.
7:N	Physical Address The size and format of this field is defined as part of the corresponding physical transport binding specification. The information given in this field is given MSB first. Any unused bits should be set to `0b`.

Step 2 – For each entry that is a single endpoint (Byte 3) and where the Physical Transport Binding (Byte 4) matches the Physical media over which the MC is trying to discover other endpoints, the MC may use the Physical Address (Byte(s) 7:N) and the EID (Byte 2) to issue the MCTP Get Message Type Support command to the endpoint.

The response from the endpoint to the Get Message Type Support command will contain the message types that are supported by the endpoint as described in Table 7. If the desired type (NC-SI over MCTP (0x02) and possibly Ethernet over MCTP (0x03) are supported, the MC may note the addressing of the endpoint (Physical Address and EID) for additional discovery and configuration to be performed later, or it could do such discovery and configuration immediately before moving to the next step.

Step 3 – Part of the data returned by the Bus Owner to the Get Routing Table Entries command includes a handle to the next Routing Table Entry. If this value is not 0xFF, the MC may continue searching for endpoints that support NC-SI over MCTP by again sending the Get Routing Table Entries MCTP command to the Bus Owner using the handle value from the previous response. A value that is equal to 0xFF is an indication that there are no additional entries. For more details, please refer to DSP0236.

7.2 Initialization of NC-SI over MCTP

NC-SI over MCTP is simply an encapsulation of the NC-SI protocol over an MCTP binding. As such, all the rules and options outlined in the NC-SI specification (DSP0222) apply to NC-SI over MCTP. This means that the MC must issue the NC-SI Clear Initial State command and follow all the steps outlined in the NC-SI specification in order to send or receive Ethernet over MCTP traffic.

7.3 Transition from SMBus to PCIe VDM

This example shows a possible flow when the MC is currently using the NC-SI over MCTP via the MCTP over SMBus binding and the PCIe interface has become available. The MC wishes to use NC-SI over MCTP via the MCTP over PCIe VDM binding for the same NC and port that it is currently using.

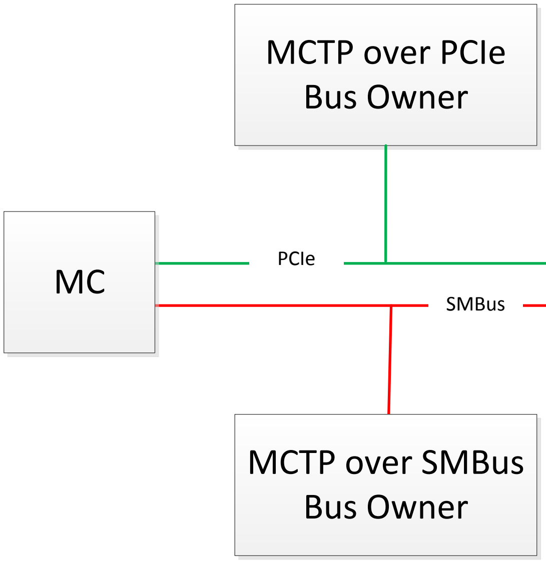


Figure 13 – Example Bus Owners

Step 1 – MC is assigned an EID by the PCIe Bus Owner when the Bus Owner issues the Set Endpoint ID command to the MC. The MC will note the Bus Owner's endpoint ID and PCIe address when it processes the Set Endpoint ID command. This is how the MC is notified that MCTP over PCIe binding is available.

Step 2 – MC issues a Get Endpoint UUID command to NC via MCTP over SMBus. Note that this step can be taken before step 1 and saved for later use.

Step 3 – MC typically "sleeps" for a while and then repeats the enumeration steps to ensure all endpoints on the MCTP over PCIe VDM Fabric have been enumerated before proceeding to step 5. Note that duration that the MC sleeps is out of the scope of the MCTP specifications.

Step 4 – MC issues the Get Routing Table Entries MCTP command to the Bus Owner via the MCTP over PCIe binding.

Step 5 – If the desired endpoint has not been found, iterate through each entry found using the Get Routing Table Entries command from step 4, where the entry type is 00b (corresponds to a single endpoint that does not operate as an MCTP bridge) issue a Get Endpoint UUID command to the endpoint via the MCTP over PCIe VDM binding. If the UUID returned matches the UUID returned from step 2, the desired endpoint has been found.

Step 6 – If the MC has configured the NC-SI over MCTP over SMBus binding to enable Ethernet traffic, or for AENs, it is recommended that the MC ensure that the endpoint is disabled from transmitting any data (either Ethernet or AENs) to the MC before moving to step 7. This is accomplished by issuing the NC-SI Disable Channel command. Issuing this command may reduce the risk of lost message fragments during the transition from one medium to another. The NC-SI Deselect Package command is a viable alternative to the Disable Channel command as well.

Step 7 –MC issues a benign NC-SI command such as Get Device ID to the NC over the MCTP over PCIe VDM binding. Recall that any NC-SI Command sent on the new media (in this case PCIe) will automatically move NC-SI and Ethernet traffic to the new media from the old (SMBus in this example).

If the MC issues the NC-SI Disable Channel command in step 6, it will need to re-enable the channel again on the new medium in order for Ethernet traffic and AENs to be received.

7.3.1.1 Alternate mechanisms

Steps 4 and 5 can be replaced with alternate mechanisms that rely on commands that are not mandatory to implement.

7.3.1.1.1 Resolve UUID method

If PCIe Bus Owner supports the MCTP Resolve UUID command, the MC can issue this command via the MCTP over PCIe binding to the Bus Owner, passing in the UUID that was obtained in step 2.

The MC can then use the data returned by the Bus Owner to determine the EID and PCIe address of the NC on the MCTP over PCIe VDM binding. Then continue with step 6 as detailed in 7.2.

Note that the Resolve UUID command is not a mandatory command for the Bus Owner to implement.

7.3.1.1.1.1 Get Supported media

The NC-SI over MCTP specification provides the optional NC-SI Get Supported Media command. The MC can issue this command to the NC via NC-SI over MCTP over the SMBus binding. The return data contains the EID and physical address information that the MC can then use to communicate with the NC via MCTP over the PCIe binding. Then continue with step 6 as detailed in 7.2.

Note that the Get Supported Media command is not a mandatory command for the endpoint to implement.

7.4 Transition from PCIe VDM to SMBus

The general scenario in which the MC might choose to transition from an MCTP over PCIe VDM binding to MCTP over SMBus binding is if the MC has determined that the PCIe interface has or will soon become unavailable (for example the system is about to be put into a low power state).

Transitioning from the NC-SI over MCTP over PCIe VDM binding to the NC-SI over MCTP over SMBus binding is accomplished by issuing any NC-SI command over the new media (SMBus).

For this example it is assumed that the MC and NC from the example detailed in clause 7.3 where the MC has already transitioned from using SMBus to using PCIe VDM.

Although unlikely to occur, it is possible for the SMBus address and EID of the NC to have changed since the MC transitioned from using SMBus to using PCIe VDM. The example flow that follows will check to see if this situation has occurred.

Step 1 – Using the last known valid addressing data for the SMBus binding (the SMBus Slave Address and EID), the MC should issue the MCTP Get Endpoint UUID command over the MCTP over SMBus binding. If the UUID that is returned matches the previously discovered UUID, it means that the EID and slave address for the NC over the SMBus binding have not changed and the MC can continue to step 2.

If, however, there is no response to the Get Endpoint UUID command, or if the UUID does not match, the MC should initiate a discovery process as outlined in clause 7.3 to discover the NC over the MCTP over SMBus binding.

Step 2 – As described in step 6 of clause 7.3, the MC should issue the NC-SI Disable Channel command over the MCTP over PCIe binding, if it is still available.

Step 3 –MC issues a benign NC-SI command, such as Get Device ID, to the NC over the MCTP over SMBus.

8 Summary

This paper was created in order to provide an introduction to MCTP packets; what they are composed of and how they are similar and different over both the SMBus/I2C and PCIe VDM physical bindings. An introduction to NC-SI over MCTP is also part of this paper and it includes several example packets over different physical medium as well.

It is the goal of this paper to have provided a gentle introduction to some of the basic concepts of MCTP as well as providing useful examples. We in the workgroup sincerely hope you find the document of use.