



1
2
3
4
5

Document Number: DSP2037

Date: 2014-02-20

Version: 1.0.0

6
7
8
9
10
11
12
13
14
15
16

MCTP Packets and NC-SI over MCTP Overview

17 **Document Type: White Paper**
18 **Document Status: DMTF Informational**
19 **Document Language: en-US**

20 Copyright notice

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

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

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37 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

43

44

CONTENTS

46	Abstract	6
47	Foreword	7
48	Acknowledgments	7
49	1 Introduction.....	9
50	1.1 Target audience	9
51	1.2 Related documents	9
52	1.3 Terminology	10
53	1.4 Acronyms and abbreviations	10
54	2 MCTP overview	11
55	2.1 Bindings	11
56	2.2 MCTP header.....	11
57	2.2.1 Endpoint IDs	12
58	2.2.2 Start of Message and End of Message bits	13
59	2.2.3 Packet Sequence Number	13
60	2.2.4 Tag Owner	13
61	2.2.5 Message Tag	14
62	2.3 Message assembly	14
63	2.4 Interleaving of MCTP messages.....	14
64	2.5 Message Payload and Message Payload header	15
65	2.5.1 MCTP Vendor Defined Messages (VDM).....	16
66	2.5.2 Example MCTP and Message Payload headers	16
67	2.6 Medium-specific header and trailer.....	18
68	2.6.1 PCIe Vendor Defined Message	18
69	2.6.2 SMBus/I2C	19
70	2.7 Bus Owner	19
71	3 Sample flows and example packets.....	19
72	3.1 Sample configuration	19
73	3.1.1 SMBus setup.....	20
74	3.1.2 PCIe setup	20
75	4 MCTP control commands.....	21
76	4.1 Sample MCTP control messages	21
77	4.1.1 MCTP Get Endpoint UUID command	21
78	4.1.2 MCTP Get Endpoint UUID command over SMBus	21
79	4.1.3 Get Endpoint UUID command over PCIe VDM	22
80	4.1.4 MCTP Get Message Type Support.....	23
81	4.1.5 MCTP Get Message Type Support command over SMBus	23
82	4.1.6 MCTP Get Message Type Support command over PCIe VDM	24
83	5 NC-SI over MCTP	26
84	5.1 NC-SI Commands over RBT packet formats.....	26
85	5.1.1 NC-SI over RBT command packet	26
86	5.1.2 NC-SI over RBT response packet	27
87	5.2 Separation of NC-SI protocol and physical binding	27
88	5.3 NC-SI over MCTP packets	27
89	5.3.1 NC-SI command over MCTP	27
90	5.3.2 NC-SI response over MCTP	28
91	5.3.3 Comparing NC-SI Command packets over RBT and over MCTP.....	28
92	5.4 NC-SI command over MCTP examples.....	29
93	5.4.1 NC-SI Clear Initial State command over MCTP over SMBus.....	29
94	5.4.2 NC-SI Clear Initial State command over MCTP over PCIe VDM	30
95	5.4.3 NC-SI Set MAC Address command over MCTP over SMBus	30
96	5.4.4 NC-SI Set MAC Address command over MCTP over PCIe VDM	32

97	6	NC-SI Pass-through (Ethernet) over MCTP	32
98	6.1	Standard Ethernet/Pass-through packets.....	32
99	6.2	NC-SI Pass-through/Ethernet over MCTP examples	34
100	6.2.1	ARP request over MCTP over SMBus	35
101	6.2.2	ARP request over MCTP over PCIe VDM	36
102	7	Example flows	36
103	7.1	Discovery of NC-SI over MCTP capable endpoints.....	36
104	7.2	Initialization of NC-SI over MCTP	38
105	7.3	Transition from SMBus to PCIe VDM	38
106	7.4	Transition from PCIe VDM to SMBus	40
107	8	Summary	41

108

109 Tables

110	Table 1 – MCTP header	12
111	Table 2 – MCTP header endpoint IDs	12
112	Table 3 – Special Endpoint IDs.....	12
113	Table 4 – MCTP flags	13
114	Table 5 – SOM and EOM combinations	13
115	Table 6 – MCTP message header	15
116	Table 7 – MCTP message types.....	15
117	Table 8 – MCTP header for an MCTP command request packet.....	16
118	Table 9 – MCTP header for an MCTP command response packet.....	16
119	Table 10 – Start Of Message example packet.....	17
120	Table 11 – Middle of message example packet.....	17
121	Table 12 – End Of Message example packet.....	17
122	Table 13 – PCIe VDM header	18
123	Table 14 – Sample MCTP over PCIe VDM packet header.....	18
124	Table 15 – PCIe VDM trailer	19
125	Table 16 – SMBus/I2C header.....	19
126	Table 17 – SMBus/I2C trailer.....	19
127	Table 18 – SMBus pins on PCIe connector	20
128	Table 19 – Get Endpoint UUID command over SMBus.....	21
129	Table 20 – Get Endpoint UUID response over SMBus.....	22
130	Table 21 – Get Endpoint UUID request over PCIe VDM	22
131	Table 22 – Get Endpoint UUID response over PCIe VDM	23
132	Table 23 – Get Message Type Support request over SMBus	24
133	Table 24 – Get Message Type Support response over SMBus	24
134	Table 25 – Get Message Type Support request over PCIe VDM.....	25
135	Table 26 – Get Message Type Support response over PCIe VDM	25
136	Table 27 – Clear Initial State command over MCTP over SMBus.....	29
137	Table 28 – Clear Initial State command over MCTP over PCIe VDM	30
138	Table 29 – Set MAC Address command over MCTP over SMBus.....	31
139	Table 30 – Set MAC Address command over MCTP over PCIe VDM	32
140	Table 31 – ARP request to MC over MCTP over SMBus	35
141	Table 32 – ARP request to MC over MCTP over PCIe VDM.....	36
142	Table 33 – Routing table entry format.....	37

143

144 **Figures**

145 Figure 1 – MCTP message encapsulation..... 11

146 Figure 2 – Sample SMBus setup for add-in card 20

147 Figure 3 – MCTP over PCIe VDM setup..... 20

148 Figure 4 – NC-SI over RBT command format..... 26

149 Figure 5 – NC-SI over RBT response format..... 27

150 Figure 6 – NC-SI command over MCTP packet 27

151 Figure 7 – NC-SI response over MCTP packet 28

152 Figure 8 – NC-SI command over RBT 28

153 Figure 9 – NC-SI command over MCTP 29

154 Figure 10 – Pass-through Ethernet packet format..... 33

155 Figure 11 – Ethernet packet over MCTP format 33

156 Figure 12 – Example MCTP topology 37

157 Figure 13 – Example Bus Owners 39

158

159

Abstract

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

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

170

Foreword

171 The *MCTP Packets and NC-SI over MCTP Overview* (DSP2037) was prepared by the PMCI Working
172 Group.

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

175 **Acknowledgments**

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

177 **Editor:**

- 178 • Patrick Kutch – Intel Corporation

179 **Contributors:**

- 180 • Alan Berenbaum – Standard Microsystems
- 181 • Patrick Caporale – IBM Corporation
- 182 • Eliel Louzoun – Intel Corporation
- 183 • Yuval Itkin – Mellanox Technologies
- 184 • Patrick Schoeller – Hewlett-Packard Company
- 185 • Hemal Shah – Broadcom Corporation
- 186 • Tom Slaight – Intel Corporation

187

189

190 **1 Introduction**

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

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

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

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

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

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

212 **1.1 Target audience**

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

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

220 **1.2 Related documents**

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

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

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

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

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

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

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

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

239 1.3 Terminology

240

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.

241 1.4 Acronyms and abbreviations

242

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	RMI Based Transport
AEN	Asynchronous Event Notification

243 **2 MCTP overview**

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

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

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

252 **2.1 Bindings**

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

- 254 • SMBus/I2C
- 255 • PCIe VDM
- 256 • KCS
- 257 • Serial

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

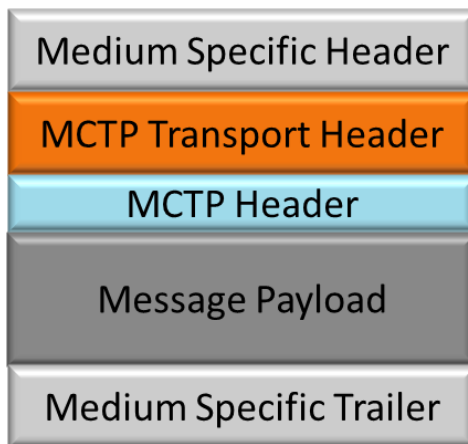


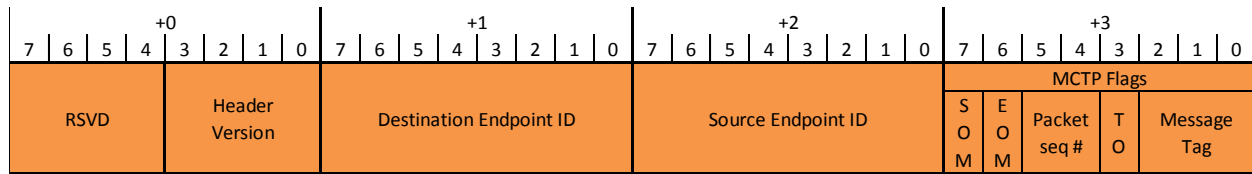
Figure 1 – MCTP message encapsulation

260 **2.2 MCTP header**

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

264

Table 1 – MCTP header



265

266

267 The definitions of these fields are detailed in [DSP0236](#).

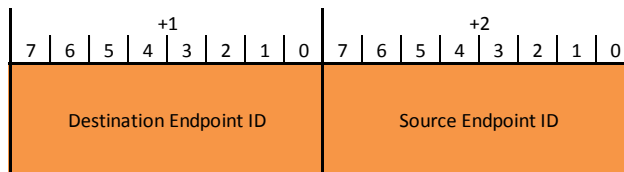
2.2.1 Endpoint IDs

269 MCTP endpoint IDs are logical numbers assigned by the [bus owner](#) to an endpoint within an MCTP
 270 network. The numbers are unique within an MCTP network.

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

274

Table 2 – MCTP header endpoint IDs



275

2.2.1.1 Special endpoint IDs

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

279

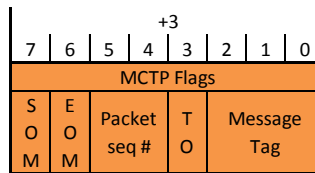
Table 3 – Special Endpoint IDs

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.

280 **2.2.2 Start of Message and End of Message bits**

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

284 **Table 4 – MCTP flags**



285
 286 The Start Of Message (SOM) and End Of Message (EOM) bits are used to indicate whether a message is
 287 the start, end, or middle of a larger message that is being transmitted in smaller chunks. Table 5 provides
 288 a description of what the different combinations of the SOM and EOM bits indicate.

289 **Table 5 – SOM and EOM combinations**

S O M	E O M	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.

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

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

296 **2.2.3 Packet Sequence Number**

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

299 **2.2.4 Tag Owner**

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

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

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

309 **2.2.5 Message Tag**

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

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

315 **2.3 Message assembly**

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

- 319 • Message Tag
- 320 • Tag Owner
- 321 • Source Endpoint ID

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

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

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

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

337 **2.4 Interleaving of MCTP messages**

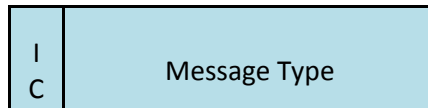
338 There are times when it may be desirable to interleave different MCTP messages. For example a Network
339 Controller may be transmitting a Ethernet Pass-through Message spanning multiple MCTP packets to the
340 Management Controller when a condition arises for which it should send an NC-SI AEN message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

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

347 **2.5 Message Payload and Message Payload header**

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

349 **Table 6 – MCTP message header**



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

353 The Message Payload header only exists if the [Start Of Message](#) bit is set. For messages that span
 354 multiple MCTP packets, only the first packet will contain the Message Payload header.

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

357 **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 identified 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 identified 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

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

360 2.5.1 MCTP Vendor Defined Messages (VDM)

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

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

367 2.5.2 Example MCTP and Message Payload headers

368 This clause provides some example MCTP and Message Payload headers.

369 2.5.2.1 Single packet message

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

371 **Table 8 – MCTP header for an MCTP command request 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																								
RSVD								Header Version 1								Destination Endpoint ID 0x12								Source Endpoint ID 0x32								MCTP Flags																							
																																S	E	Packet				T	Message																
																																O	O	seq #				O	Tag																
																																M	M	0					3																
																																1	1	0				1	3																
IC	Message Type																															0	0 (MCTP Control)																						
0																																																							

372
373 This message is transmitted over a single MCTP packet. It came from EID 0x12 and is being sent to EID
374 0x32. Both of the SOM (Start Of Message) and EOM (End Of Message) bits are set (indicating the
375 complete message is contained within and it does not span multiple packets). The Tag Owner bit is also
376 set, indicating that it was the source EID that determined the Message Tag. Note that the MCTP Message
377 Type field is also present as the SOM bit is set. This particular example request (and subsequent
378 response) is an MCTP Control messages.

379 **Table 9 – MCTP header for an MCTP command response 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																																
RSVD								Header Version 1								Destination Endpoint ID 0x32								Source Endpoint ID 0x12								MCTP Flags																															
																																S	E	Packet				T	Message																								
																																O	O	seq #				O	Tag																								
																																M	M	0				0	3																								
																																1	1	0				0	3																								
IC	Message Type																																							0	0 (MCTP Control)																						
0																																																															

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

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

387 **2.5.2.2 Multipacket message**

388 **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
RSVD				Header Version				Destination Endpoint ID				Source Endpoint ID				MCTP Flags															
				1				0x32				0x12				SOM	EOM	Packet seq #	TO	Message Tag											
				1				0				0				1	0	0	1	6											
IC	Message Type																														
0	3 (Ethernet)																														

389
 390 Table 10 is an MCTP Packet that is the beginning of a multipacket message. The SOM bit is set and the
 391 EOM bit is clear, indicating that it is the first Packet and more are to follow. The Message Type field is
 392 present (because the SOM bit is set) and indicates that this is an Ethernet message. The Packet
 393 Sequence Number is 0, and the Tag Owner bit is set to show that the source EID is the one that
 394 configured the Message Tag to be 6.

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

397 **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
RSVD				Header Version				Destination Endpoint ID				Source Endpoint ID				MCTP Flags															
				1				0x32				0x12				SOM	EOM	Packet seq #	TO	Message Tag											
				1				0				0				0	0	1	1	6											

398
 399 Table 11 is the second packet in this multipacket message example. Both the SOM and EOM bits are
 400 cleared, indicating that this packet is a middle packet, neither the start nor the end of the message.
 401 Because the SOM bit is cleared, the Message Type field is not present. The Tag Owner bit is still set
 402 because the source EID is the one that configured the Message Tag to be 6.

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

406 **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
RSVD				Header Version				Destination Endpoint ID				Source Endpoint ID				MCTP Flags															
				1				0x32				0x12				SOM	EOM	Packet seq #	TO	Message Tag											
				1				0				0				0	1	2	1	6											

407
 408 Table 12 shows the MCTP header of the last packet in a multipacket message. The SOM bit is cleared
 409 while the EOM bit is set, indicating that this is the last packet for the message. The Packet Sequence
 410 Number has been incremented by one from the previous example. The Tag Owner and Message Tag
 411 fields remain the same as the previous packets because they are constant for the entire message that
 412 was split into three packets.

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

442

Table 15 – PCIe VDM trailer

7	6	5	4	3	2	1	0
PCIe Padding 0x00							

443

444 **2.6.2 SMBus/I2C**

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

448

Table 16 – SMBus/I2C header

	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 1	Destination Slave Address +0								Command Code 0x0F +1								Byte Count +2								Source Slave Address +3							

449

450

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

454

Table 17 – SMBus/I2C trailer

7	6	5	4	3	2	1	0
PEC							

455

456 **2.7 Bus Owner**

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

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

463 **3 Sample flows and example packets**

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

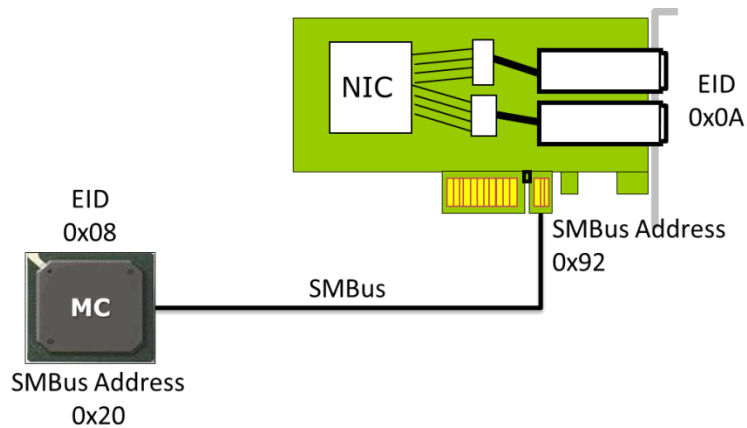
467 The remaining clauses detail sample flows that the MC might perform. Each sample shows the MCTP
 468 packet over both SMBus and PCIe VDM.

469 **3.1 Sample configuration**

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

472 **3.1.1 SMBus setup**

473 The configuration for SMBus is shown in Figure 2.



474

475 **Figure 2 – Sample SMBus setup for add-in card**

476 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:

479

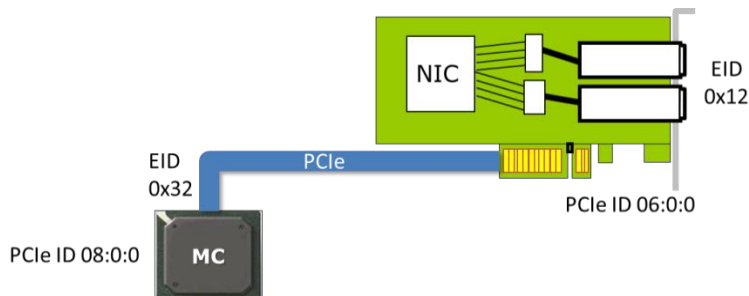
Table 18 – SMBus pins on PCIe connector

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

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

481 **3.1.2 PCIe setup**

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



483

484 **Figure 3 – MCTP over PCIe VDM setup**

485 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
 486 MC is connected to an endpoint (an NIC) with a PCIe ID of 06:0:0 and was assigned an EID of 0x12 by
 487 the Bus Owner.

488 **4 MCTP control commands**

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

492 **4.1 Sample MCTP control messages**

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

496 **4.1.1 MCTP Get Endpoint UUID command**

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

500 **4.1.2 MCTP Get Endpoint UUID command over SMBus**

501 **4.1.2.1 Request**

502

503

Table 19 – Get Endpoint UUID command over SMBus

	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							
	S		E		Packet		T		Message																							
	O		O		seq #		O		Tag																							
	M		M		3		1		3																							
Byte 8	IC	Message Type 0 (MCTP Control)																														
Byte 9	Rq	D	Instance ID 25													Command Code 03 (Get Endpoint UUID)																
	1	0	R																													
Byte 11	PEC 0x89																															

504

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

508 All MCTP control commands fit within a single packet, as such the [Start of Message](#) and End of Message
 509 bits are set.

510 4.1.2.2 Response

511 Table 20 – Get Endpoint UUID response over SMBus

	+0								+1								+2								+3									
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									
																									S O M		E O M		Packet seq #		T O		Message Tag	
																									1		1		0		0		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																																	

512

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

516 4.1.3 Get Endpoint UUID command over PCIe VDM

517 4.1.3.1 Request

518 Table 21 – Get Endpoint UUID request over PCIe VDM

	+0								+1								+2								+3									
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				Message Code Vendor Defined 0111 1111b																	
	R		Pad Len 1		MCTP VDM Code (0000b)																													
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									
																									S O M		E O M		Packet seq #		T O		Message Tag	
																									1		1		3		1		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																																	

519

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

- 522 • Start Of Message bit
- 523 • End Of Message Bit
- 524 • Tag Owner bit
- 525 • Message Type field
- 526 • MCTP command request bit
- 527 • MCTP command code

528 **4.1.3.2 Response**

529 **Table 22 – Get Endpoint UUID response over PCIe VDM**

	+0								+1								+2								+3																																						
Byte 0	R	7	6	5	4	3	2	1	0	R	7	6	5	4	3	2	1	0	TD	7	6	5	4	3	2	1	0	EP	7	6	5	4	3	2	1	0	Attr	7	6	5	4	3	2	1	0	R	7	6	5	4	3	2	1	0	Length	7	6	5	4	3	2	1	0
	Fmt 11								Type 10 r2r10								TC 000								Length 5																																						
Byte 4	PCIe Requester ID 0600																PCIe TAG Field				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																																														
													S O M				E O M				Packet seq #				T O				Message Tag																																		
Byte 16	IC	7	6	5	4	3	2	1	0	Message Type 0 (MCTP Control)																																																					
Byte 17	Rq	D	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																																																		

531 The response to the MCTP Get Endpoint UUID over PCIe VDM is detailed in Table 22. Note that the
 532 actual MCTP response data is identical to the data received for the same response over SMBus in Table
 533 20 with the exception of the instance ID.

534 **4.1.4 MCTP Get Message Type Support**

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

538 **4.1.5 MCTP Get Message Type Support command over SMBus**

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

541 **4.1.5.1 Request**

542 **Table 23 – Get Message Type Support request over SMBus**

	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										
																									S	E	Packet				T	Message			
																									O	O	seq #				O	Tag			
																									M	M	3					3			
Byte 8	IC	Message Type 0 (MCTP Control)																																	
Byte 9	Rq	D	R	Instance ID 26				Command Code 05 (Get Message Type Support)																											
Byte 11	PEC 0xA4																																		

544 **4.1.5.2 Response**

545 **Table 24 – Get Message Type Support response over SMBus**

	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										
																									S	E	Packet				T	Message			
																									O	O	seq #				O	Tag			
																									M	M	0					0			
Byte 8	IC	Message Type 0 (MCTP Control)																																	
Byte 9	Rq	D	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																																		

547 As Table 24 shows, the endpoint (a NIC) responded to the MCTP Get Message Type Support
 548 request successfully. The returned data indicates that this endpoint supports two message types. Types
 549 0x02 and 0x03 are indicated as being supported, which according to Table 7 correspond to NC-SI
 550 Command and NC-SI Pass-through (Ethernet) types.

551 **4.1.6 MCTP Get Message Type Support command over PCIe VDM**

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

554 **4.1.6.1 Request**

555 **Table 25 – Get Message Type Support request over PCIe VDM**

Byte 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	7	6	5	4	3	2	1	0
	R	Fmt 11			Type 10 r2r10				R	TC 000				R				TD	EP	Attr 01		R	Length 1									
Byte 4	PCIe Requester ID 0800										PCIe TAG Field R				MCTP VDM Code 1 (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																	
															S	E	Packet	T	Message													
															O	O	seq #	O	Tag													
															M	M																
															1	1	3	1	3													
Byte 16	IC	Message Type 0 (MCTP Control)																														
Byte 17	Rq	D	Instance ID 14				Command Code 05 (Get Message Type Support)																									
	1	0																														
Byte 19	PCIe Padding 0x00																															

556

557 **4.1.6.2 Response**

558 **Table 26 – Get Message Type Support response over PCIe VDM**

Byte 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	7	6	5	4	3	2	1	0
	R	Fmt 11			Type 10 r2r10				R	TC 000				R				TD	EP	Attr 01		R	Length 2									
Byte 4	PCIe Requester ID 0600										PCIe TAG Field R				MCTP VDM Code 0 (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																	
															S	E	Packet	T	Message													
															O	O	seq #	O	Tag													
															M	M																
															1	1	0	0	3													
Byte 16	IC	Message Type 0 (MCTP Control)																														
Byte 17	Rq	D	Instance ID 14				Command Code 05 (Get Message Type Support)				Completion Code 00				Data 0x03																	
	0	0																														
Byte 21	Data 0x02																															
	Data 0x03																															
	Data 0x7E																															

559

560 As Table 26 shows, the endpoint (a NIC) responded to the MCTP Get Message Type Support
 561 request successfully. The returned data indicates that this endpoint supports three message types. Types
 562 0x02, 0x03, and 0x7E are indicated as being supported, which according to Table 7 correspond to NC-SI
 563 Command, NC-SI Pass-through (Ethernet), and Vendor Defined - PCIe types.

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

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

579 **5 NC-SI over MCTP**

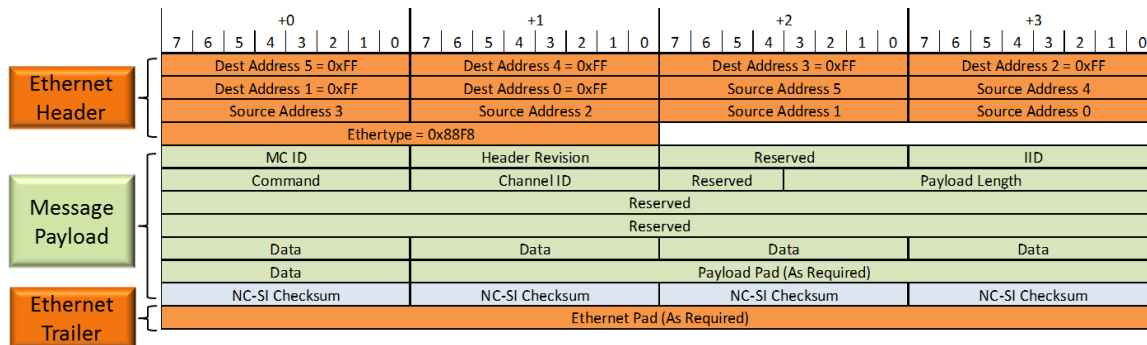
570 The *NC-SI over MCTP Binding Specification* ([DSP0261](#)) details how to package NC-SI Commands over
 571 MCTP in addition to how to package NC-SI Pass-Through Ethernet traffic. This clause provides an
 572 overview of how NC-SI Commands over MCTP are packaged.

573 **5.1 NC-SI Commands over RBT packet formats**

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

576 **5.1.1 NC-SI over RBT command packet**

577



578

Figure 4 – NC-SI over RBT command format

579

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

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

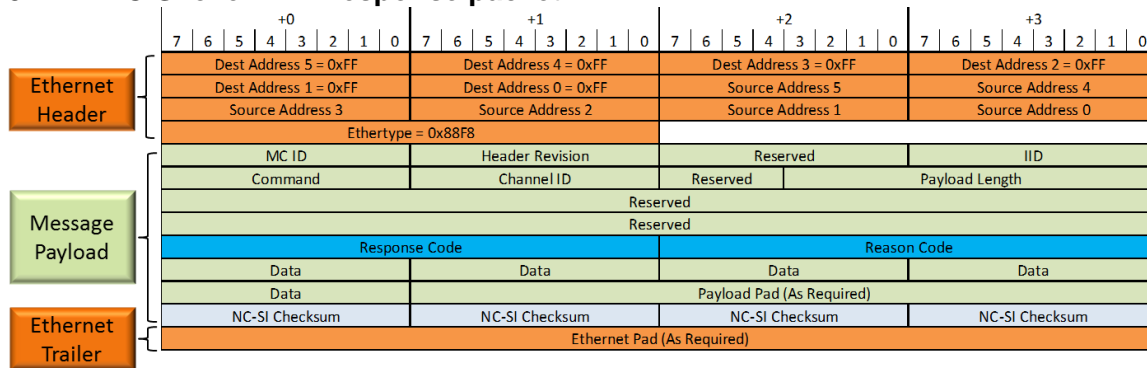
585 The Ethernet header comprises:

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

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

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

592 **5.1.2 NC-SI over RBT response packet**



593 **Figure 5 – NC-SI over RBT response format**

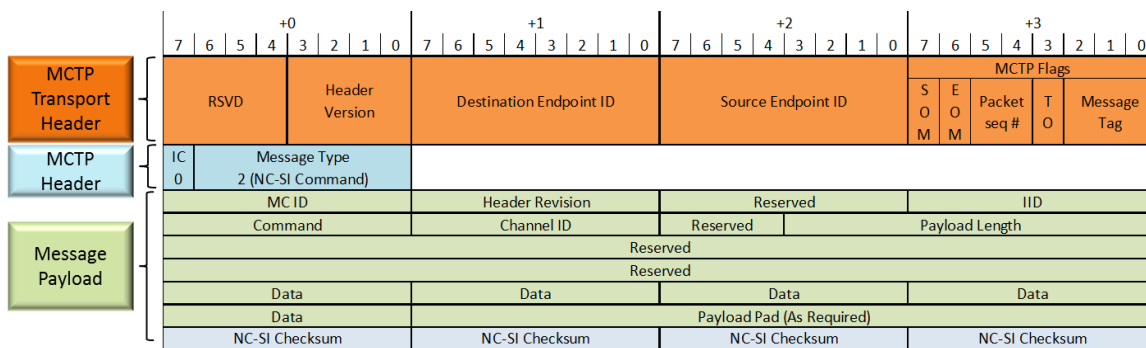
594
595 Figure 5 shows the packet format of a NC-SI over RBT response. It has all the same fields and
596 requirements as the NC-SI over RBT command as discussed in 5.1.1. NC-SI responses contain two
597 word-sized fields that are not present in the NC-SI Command: the response and reason codes.

598 **5.2 Separation of NC-SI protocol and physical binding**

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

606 **5.3 NC-SI over MCTP packets**

607 **5.3.1 NC-SI command over MCTP**

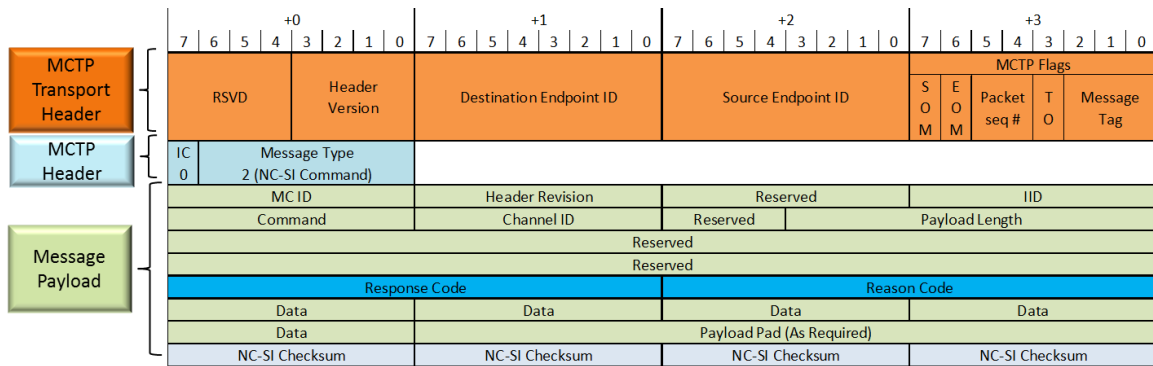


608 **Figure 6 – NC-SI command over MCTP packet**

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

- 612 • MCTP transport header (see 2.2)
- 613 • MCTP header (see 2.5)
- 614 • MCTP command (the payload)

615 **5.3.2 NC-SI response over MCTP**



616

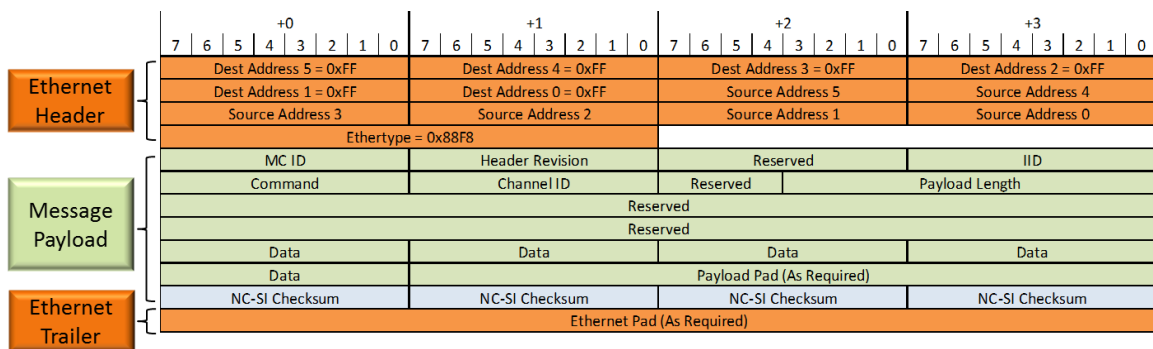
617 **Figure 7 – NC-SI response over MCTP packet**

618 As you can see in Figure 7, a NC-SI Response of MCTP has the same components as an [NC-SI](#)
 619 [Command over MCTP](#):

- 620 • MCTP transport header (see 2.2)
- 621 • MCTP header (see 2.5)
- 622 • MCTP command response (the payload)

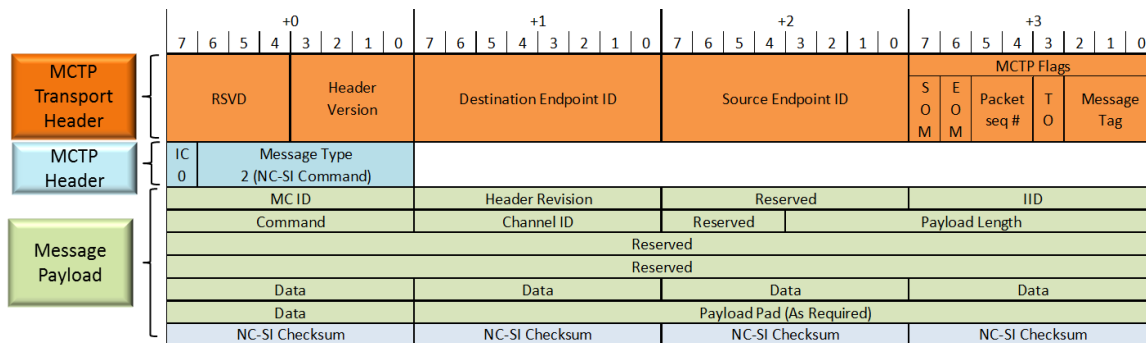
623 In the case of the response, the response and reason codes are present within the NC-SI response.

624 **5.3.3 Comparing NC-SI Command packets over RBT and over MCTP**



625

626 **Figure 8 – NC-SI command over RBT**



627
628

Figure 9 – NC-SI command over MCTP

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

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

635 **5.4 NC-SI command over MCTP examples**

636 **5.4.1 NC-SI Clear Initial State command over MCTP over SMBus**

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

640 **5.4.1.1 Request**

641 **Table 27 – Clear Initial State command over MCTP over SMBus**

	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 26				Source Slave Address 0x21																			
Byte 4	RSVD		Header Version 1		Destination Endpoint ID 0x0A				Source Endpoint ID 0x08				MCTP Flags S O M: 1, E O M: 1, Packet seq #: 3, T O: 1, Message Tag: 3																			
Byte 8	IC 0	Message Type 2 (NC-SI Command)						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																															

642

643 Just as in the [MCTP Control examples](#), note how the Start of Message and End Of Message bits are set,
644 indicating that this message is contained within a single packet. Also note that the NC-SI checksum fields

645 are present within the NC-SI command. This is because those fields are part of the NC-SI message itself
 646 (which remains wholly intact in NC-SI over MCTP).

647 **5.4.2 NC-SI Clear Initial State command over MCTP over PCIe VDM**

648 **5.4.2.1 Request**

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

	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 6									
Byte 4	PCIe Requester ID 0800								PCIe TAG Field				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																			
Byte 16	IC 0	Message Type 2 (NC-SI Command)																														
Byte 17	MC ID 0x00 (Clear Initial State)				Header Revision 0x01				Reserved				IID 81																			
Byte 21	Command 0x00				Channel ID 0x00				Reserved				Payload Length 0																			
Byte 25	Reserved																															
Byte 29	Reserved																															
Byte 33	NC-SI Checksum 0x00				NC-SI Checksum 0x00				NC-SI Checksum 0x00				NC-SI Checksum 0x00																			
Byte 37	PCIe Padding 0x00				PCIe Padding 0x00				PCIe Padding 0x00																							

650

651

652 Although the transport and MCTP headers differ from the same command being sent over SMBus (see
 653 5.4.1.1), the actual NC-SI Command in the MCTP Payload is, with the exception of the IID, exactly the
 654 same. Remember that MCTP provides a mechanism to send a payload over different mediums.

655 **5.4.3 NC-SI Set MAC Address command over MCTP over SMBus**

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

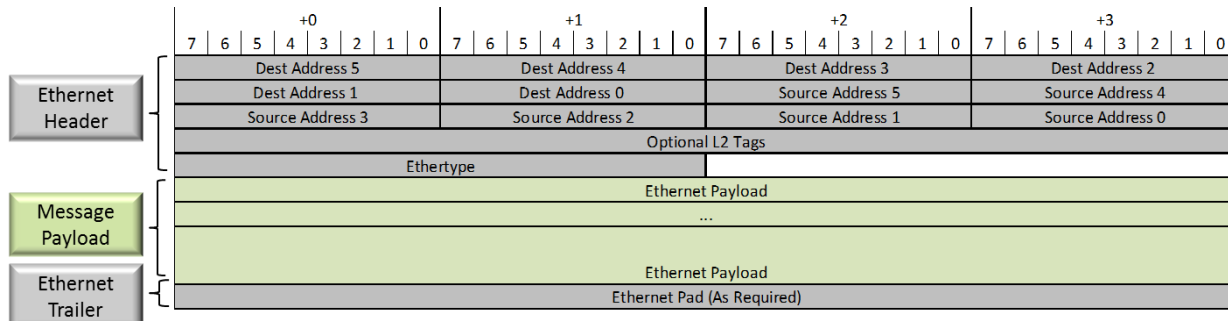
658 5.4.3.1 Request

659 Table 29 – Set MAC Address command over MCTP over SMBus

	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 34								Source Slave Address 0x21										
Byte 4	RSVD				Header Version 1				Destination Endpoint ID 0x0A								Source Endpoint ID 0x08								MCTP Flags										
																									S	E	Packet				T	Message			
																									O	O	seq #				O	Tag			
																									M	M	3					3			
Byte 8	IC	Message Type 2 (NC-SI Command)																																	
Byte 9	MC ID 0x00								Header Revision 0x01								Reserved								IID 86										
Byte 13	Command 0x0E (Set MAC Address)								Channel ID 0x00								Reserved								Payload Length 8										
Byte 17	Reserved																																		
Byte 21	Reserved																																		
Byte 25	Data 0x00								Data 0x25								Data 0x90								Data 0x7E										
Byte 29	Data 0x91								Data 0xE5								Data 0x01								Data 0x01										
Byte 33	NC-SI Checksum 0x00								NC-SI Checksum 0x00								NC-SI Checksum 0x00								NC-SI Checksum 0x00										
Byte 37	PEC 0x1A																																		

660

661



679

Figure 10 – Pass-through Ethernet packet format

680

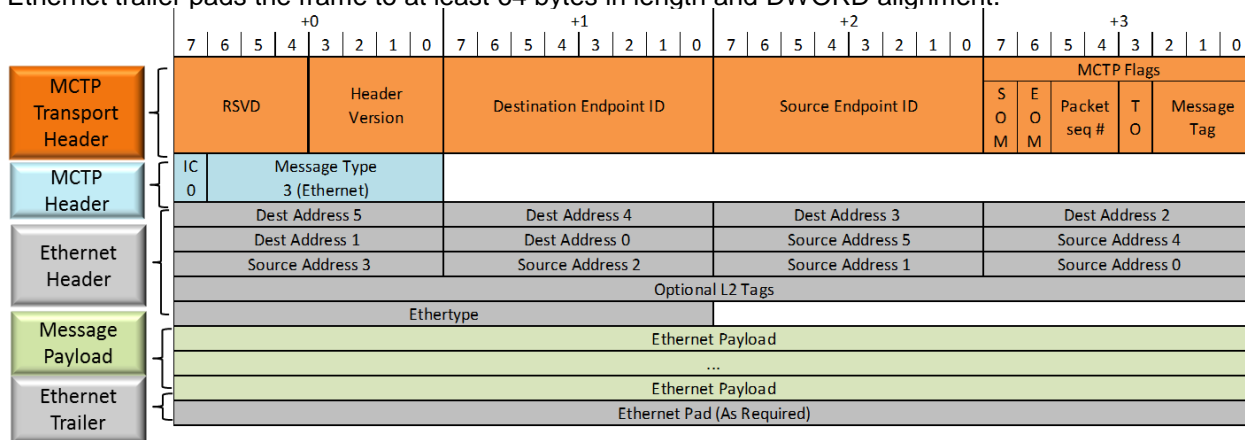
681

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

- 683 • Destination address
- 684 • Source address
- 685 • Optional L2 tags
- 686 • EtherType
- 687 • Payload
- 688 • Ethernet pad

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

692



693

Figure 11 – Ethernet packet over MCTP format

694

695 Figure 11 details an Ethernet packet (NC-SI Pass-through) over MCTP. As with all MCTP packets, it
 696 contains the MCTP header. According to Table 7, the MCTP header indicates that the payload is an
 697 Ethernet frame. The actual Ethernet data format is identical to a standard Ethernet frame, as shown in
 698 6.1, containing:

- 699 • Destination address
- 700 • Source address

- 701 • Optional L2 tags
- 702 • EtherType
- 703 • Payload
- 704 • Ethernet pad

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

708 **6.2 NC-SI Pass-through/Ethernet over MCTP examples**

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

712 **6.2.1 ARP request over MCTP over SMBus**

713 **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									
																									S O M		E O M		Packet seq #		T O		Message Tag	
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																																	

714

715 **6.2.2 ARP request over MCTP over PCIe VDM**

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

717

	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																	
															S O M	E O M	Packet seq #	T O	Message Tag													
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																				

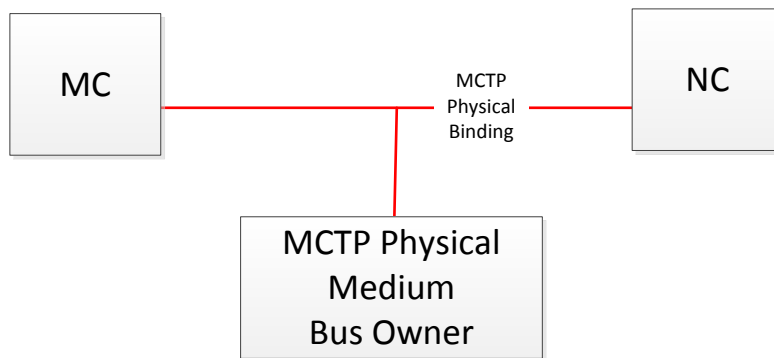
718

719 **7 Example flows**

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

721 **7.1 Discovery of NC-SI over MCTP capable endpoints**

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



724
725

Figure 12 – Example MCTP topology

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

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

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

733

Table 33 – Routing table entry format

Byte	Description
1	Size of EID range associated with this entry
2	Starting EID
3	<p>Entry Type/Port Number</p> <p>[7:6] – Entry Type:</p> <ul style="list-style-type: none"> 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. <p>[5] – Dynamic/Static Entry</p> <p>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.</p> <ul style="list-style-type: none"> 0b = Entry was dynamically created. 1b = Entry was statically configured. <p>[4:0] – Port Number</p> <p>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.</p> <p>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</p>

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.

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

738 The response from the endpoint to the Get Message Type Support command will contain the message
739 types that are supported by the endpoint as described in Table 7. If the desired type (NC-SI over MCTP
740 (0x02) and possibly Ethernet over MCTP (0x03) are supported, the MC may note the addressing of the
741 endpoint (Physical Address and EID) for additional discovery and configuration to be performed later, or it
742 could do such discovery and configuration immediately before moving to the next step.

743 Step 3 – Part of the data returned by the Bus Owner to the Get Routing Table Entries command includes
744 a handle to the next Routing Table Entry. If this value is not 0xFF, the MC may continue searching for
745 endpoints that support NC-SI over MCTP by again sending the Get Routing Table Entries MCTP
746 command to the Bus Owner using the handle value from the previous response. A value that is equal to
747 0xFF is an indication that there are no additional entries. For more details, please refer to [DSP0236](#).

748 7.2 Initialization of NC-SI over MCTP

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

753 7.3 Transition from SMBus to PCIe VDM

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

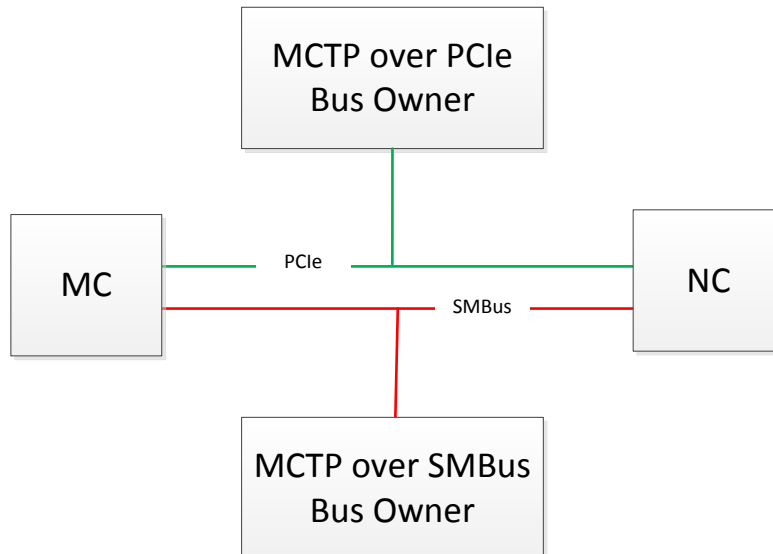


Figure 13 – Example Bus Owners

757

758

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

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

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

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

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

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

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

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

785 7.3.1.1 Alternate mechanisms

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

788 7.3.1.1.1 Resolve UUID method

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

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

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

794 7.3.1.1.1.1 Get Supported media

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

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

801 7.4 Transition from PCIe VDM to SMBus

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

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

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

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

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

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

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

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

823 **8 Summary**

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

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