



Document Identifier: DSP0236

Date: 2016-11-24

Version: 1.3.0

1
2
3
4

5 **Management Component Transport Protocol**
6 **(MCTP) Base Specification**
7 **Includes MCTP Control Specifications**

8 **Supersedes: 1.2.1**

9 **Document Class: Normative**

10 **Document Status: Published**

11 **Document Language: en-US**

12 Copyright notice

13 Copyright © 2016 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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29 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

35 PCI-SIG, PCIe, and the PCI HOT PLUG design mark are registered trademarks or service marks of PCI-
36 SIG.

37 All other marks and brands are the property of their respective owners.

CONTENTS

39 Foreword 6

40 Introduction..... 7

41 1 Scope 9

42 2 Normative references 9

43 2.1 Approved references..... 9

44 2.2 Other references 9

45 3 Terms and definitions 10

46 3.1 Requirement term definitions 10

47 3.2 MCTP term definitions 12

48 4 Symbols and abbreviated terms..... 18

49 5 Conventions 21

50 5.1 Byte ordering..... 21

51 5.2 Reserved fields 21

52 6 Management component relationships 21

53 7 MCTP overview 21

54 8 MCTP base protocol..... 25

55 8.1 MCTP packet fields..... 25

56 8.2 Special endpoint IDs 27

57 8.3 Packet payload and transmission unit sizes 28

58 8.4 Maximum message body sizes..... 28

59 8.5 Message assembly 28

60 8.6 Dropped packets..... 29

61 8.7 Starting message assembly..... 29

62 8.8 Terminating message assembly/dropped messages 29

63 8.9 Dropped messages..... 30

64 8.10 MCTP versioning and message type support..... 31

65 8.11 MCTP message types..... 32

66 8.12 Security 32

67 8.13 Limitations 32

68 8.14 MCTP discovery and addressing..... 33

69 8.15 Devices with multiple media interfaces..... 34

70 8.16 Peer transactions 34

71 8.17 Endpoint ID assignment and endpoint ID pools 34

72 8.18 Handling reassigned EIDs 39

73 9 MCTP bridging 40

74 9.2 Bridge and routing table examples 48

75 9.3 Endpoint ID resolution..... 52

76 9.4 Bridge and bus owner implementation recommendations..... 54

77 9.5 Path and transmission unit discovery 55

78 9.6 Path transmission unit requirements for bridges 58

79 10 Rate limiting..... 59

80 11 MCTP control protocol..... 62

81 11.1 Terminology 62

82 11.2 MCTP control message format 63

83 11.3 MCTP control message fields 64

84 11.4 MCTP control message transmission unit size 65

85 11.5 Tag Owner (TO), Request (Rq), and Datagram (D) bit usage..... 65

86 11.6 Concurrent command processing..... 66

87 12 MCTP control messages 66

88 12.1 MCTP control message command codes..... 66

89 12.2 MCTP control message completion codes 69

90	12.3	Set Endpoint ID.....	69
91	12.4	Get Endpoint ID	71
92	12.5	Get Endpoint UUID	72
93	12.6	Get MCTP version support	73
94	12.7	Get Message Type Support.....	76
95	12.8	Get Vendor Defined Message Support.....	76
96	12.9	Resolve Endpoint ID	78
97	12.10	Allocate Endpoint IDs.....	79
98	12.11	Routing Information Update.....	81
99	12.12	Get Routing Table Entries	83
100	12.13	Prepare for Endpoint Discovery.....	84
101	12.14	Endpoint Discovery	85
102	12.15	Discovery Notify	85
103	12.16	Get Network ID	86
104	12.17	Query Hop.....	86
105	12.18	Resolve UUID	87
106	12.19	Query rate limit.....	88
107	12.20	Request TX rate limit	89
108	12.21	Update rate limit.....	90
109	12.22	Query supported interfaces	90
110	12.23	Transport Specific	91
111	13	Vendor Defined – PCI and Vendor Defined – IANA messages	91
112	13.1	Vendor Defined – PCI message format	92
113	13.2	Vendor Defined – IANA message format.....	92
114		ANNEX A (informative) Notation	93
115		ANNEX B (informative) Change log.....	94
116			

117 Figures

118	Figure 1 – Management component relationships.....	21
119	Figure 2 – MCTP networks	22
120	Figure 3 – MCTP topology	24
121	Figure 4 – Generic message fields	25
122	Figure 5 – Topmost bus owners	35
123	Figure 6 – Split bridge	36
124	Figure 7 – Acceptable failover/redundant communication topologies	41
125	Figure 8 – Routing/bridging restrictions	41
126	Figure 9 – EID options for MCTP bridges	42
127	Figure 10 – Basic routing table entry fields.....	45
128	Figure 11 – Routing table population	46
129	Figure 12 – Example 1 Routing topology.....	48
130	Figure 13 – Example 2 Routing topology.....	50
131	Figure 14 – Example 3 Routing topology.....	51
132	Figure 15 – Endpoint ID resolution	53
133	Figure 16 – Resolving multiple paths.....	54
134	Figure 17 – Example path routing topology	56
135	Figure 18 – Path transmission unit discovery flowchart.....	58
136	Figure 19 – Example rate limiting message exchanges	60
137	Figure 20 – MCTP control message format.....	64
138	Figure 21 – Structure of Vendor ID field for Get Vendor Defined capabilities message	77

139 Figure 22 – EID Pools from multiple bus owners 80

140 **Tables**

141 Table 1 – MCTP base protocol common fields 25

142 Table 2 – Special endpoint IDs 27

143 Table 3 – MCTP Message Types Used in this Specification 32

144 Table 4 – Example 1 Routing table for D2 49

145 Table 5 – Example 2 Routing table for D1 50

146 Table 6 – Example 3 Routing table for D2 51

147 Table 7 – Additional information tracked by bridges 52

148 Table 8 – MCTP control protocol terminology 63

149 Table 9 – MCTP control message types 63

150 Table 10 – MCTP control message fields 64

151 Table 11 – Tag Owner (TO), Request (Rq) and Datagram (D) bit usage 65

152 Table 12 – MCTP control command numbers 67

153 Table 13 – MCTP control message completion codes 69

154 Table 14 – Set Endpoint ID message 70

155 Table 15 – Get Endpoint ID message 71

156 Table 16 – Get Endpoint UUID message format 72

157 Table 17 – Example UUID format 73

158 Table 18 – Get MCTP version support message 73

159 Table 19 – Get Message Type Support message 76

160 Table 20 – Get Vendor Defined Message Support message 77

161 Table 21 – Vendor ID formats 78

162 Table 22 – Resolve Endpoint ID message 78

163 Table 23 – Allocate Endpoint IDs message 80

164 Table 24 – Routing Information Update message 82

165 Table 25 – Routing Information Update entry format 82

166 Table 26 – Get Routing Table Entries message 83

167 Table 27 – Routing Table Entry format 83

168 Table 28 – Prepare for Endpoint Discovery message 85

169 Table 29 – Endpoint Discovery message 85

170 Table 30 – Discovery Notify message 85

171 Table 31 – Get Network ID message format 86

172 Table 32 – Query Hop message 86

173 Table 33 – Resolve UUID message 87

174 Table 34 – Resolve UUID message entry format 88

175 Table 35 – Query rate limit message 88

176 Table 36 – Request TX rate limit message 89

177 Table 37 – Update rate limit message 90

178 Table 38 – Query supported interfaces 90

179 Table 39 – Transport Specific message 91

180 Table 40 – Vendor Defined – PCI message format 92

181 Table 41 – Vendor Defined – IANA message format 92

182

183

Foreword

184 The *Management Component Transport Protocol (MCTP) Base Specification* (DSP0236) was prepared
185 by the PMCI Working Group.

186 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
187 management and interoperability.

188

Introduction

189 The Management Component Transport Protocol (MCTP) defines a communication model intended to
190 facilitate communication between:

- 191 • Management controllers and other management controllers
- 192 • Management controllers and managed devices

193 The communication model includes a message format, transport description, message exchange
194 patterns, and configuration and initialization messages.

195 MCTP is designed so that it can potentially be used on many bus types. The protocol is intended to be
196 used for intercommunication between elements of platform management subsystems used in computer
197 systems, and is suitable for use in mobile, desktop, workstation, and server platforms. Management
198 controllers such as a baseboard management controller (BMC) can use this protocol for communication
199 between one another, as well as for accessing managed devices within the platform.

200 Management controllers can use this protocol to send and receive MCTP-formatted messages across the
201 different bus types that are used to access managed devices and other management controllers.
202 Managed devices in a system need to provide an implementation of the message format to facilitate
203 actions performed by management controllers.

204 It is intended that different types of devices in a management system may need to implement different
205 portions of the complete capabilities defined by this protocol. Where relevant, this is called out in the
206 individual requirements
207

208 Management Component Transport Protocol (MCTP) Base 209 Specification

210 1 Scope

211 The *MCTP Base Specification* describes the command protocol, requirements, and use cases of a
212 transport protocol for communication between discrete management controllers on a platform, as well as
213 between management controllers and the devices they manage.

214 This document is intended to meet the following objectives:

- 215 • Describe the MCTP Base transport protocol
- 216 • Describe the MCTP control message protocol

217 The MCTP specifies a transport protocol format. This protocol is independent of the underlying physical
218 bus properties, as well as the "data-link" layer messaging used on the bus. The physical and data-link
219 layer methods for MCTP communication across a given medium are defined by companion "transport
220 binding" specifications, such as [DSP0238](#), MCTP over PCIe® Vendor Defined Messaging, and [DSP0237](#),
221 MCTP over SMBus/I²C. This approach enables future transport bindings to be defined to support
222 additional buses such as USB, RMII, and others, without affecting the base MCTP specification.

223 2 Normative references

224 The following referenced documents are indispensable for the application of this document. For dated
225 references, only the edition cited applies. For undated references, the latest edition of the referenced
226 document (including any amendments) applies.

227 2.1 Approved references

228 DMTF DSP4004, *DMTF Release Process v2.7*

229 http://www.dmtf.org/standards/published_documents/DSP4004_2.7.x.pdf

230 DMTF DSP2016, Management Component Transport Protocol (MCTP) Overview White Paper

231 http://www.dmtf.org/standards/published_documents/DSP2016.pdf

232 DMTF, DSP0239, *Management Component Transport Protocol (MCTP) IDs and Codes*

233 http://www.dmtf.org/standards/published_documents/DSP0239_1.3.x.pdf

234 DMTF DSP0237, Management Component Transport Protocol SMBus/I2C Transport Binding
235 Specification

236 http://www.dmtf.org/standards/published_documents/DSP0237_1.0.x.pdf

237 DMTF DSP0238, Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding
238 Specification

239 http://www.dmtf.org/standards/published_documents/DSP0238_1.0.x.pdf

240 2.2 Other references

241 Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba, *Advanced Configuration and Power Interface*
242 *Specification v5.0*, ACPI, December 6, 2011

243 <http://www.acpi.info/downloads/ACPIspec50.pdf>

- 244 IETF, RFC20, *ASCII format for Network Interchange*, October 16, 1969
245 <http://tools.ietf.org/html/rfc20>
- 246 IETF, RFC4122, *A Universally Unique Identifier (UUID) URN Namespace*, July 2005
247 <http://datatracker.ietf.org/doc/rfc4122/>
- 248 IETF, RFC2119, *Key Words for use in RFCs to Indicate Requirement Levels*, March 1997
249 <http://datatracker.ietf.org/doc/rfc2119/>
- 250 Intel, Hewlett-Packard, NEC, and Dell, *Intelligent Platform Management Interface Specification: Second Generation v2.0*, IPMI, 2004
251
252 <http://www.intel.com/design/servers/ipmi>
- 253 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*
254 <http://isotc.iso.org/livelink/livelink?func=ll&objId=4230456&objAction=browse&sort=subtype>
- 255 PCI-SIG, PCI Express™ Specifications
256 <http://www.pcisig.com/specifications/pciexpress/>
- 257 NXP Semiconductors, *UM10204 I2C-bus specification and user manual*, Rev. 5, October 9, 2012
258 http://www.nxp.com/documents/user_manual/UM10204.pdf
- 259 SMBus, *System Management Bus (SMBus) Specification v2.0*, SMBus, 2000
260 <http://www.smbus.org/specs/smbus20.pdf>

261 **3 Terms and definitions**

262 For the purposes of this document, the following terms and definitions apply.

263 **3.1 Requirement term definitions**

264 This clause defines key phrases and words that denote requirement levels in this specification. These
265 definitions are consistent with the terms defined in [RFC2119](#).

266 **3.1.1**

267 **can**

268 used for statements of possibility and capability, whether material, physical, or causal

269 **3.1.2**

270 **cannot**

271 used for statements of possibility and capability, whether material, physical or causal

272 **3.1.3**

273 **conditional**

274 indicates requirements to be followed strictly to conform to the document when the specified conditions
275 are met

276 **3.1.4**

277 **deprecated**

278 indicates that an element or profile behavior has been outdated by newer constructs

- 279 **3.1.5**
280 **mandatory**
281 indicates requirements to be followed strictly to conform to the document and from which no deviation is
282 permitted
- 283 **3.1.6**
284 **may**
285 indicates a course of action permissible within the limits of the document
- 286 NOTE: An implementation that does *not* include a particular option shall be prepared to interoperate with another
287 implementation that *does* include the option, although perhaps with reduced functionality. An implementation that
288 *does* include a particular option shall be prepared to interoperate with another implementation that does *not* include
289 the option (except for the feature that the option provides).
- 290 **3.1.7**
291 **may not**
292 indicates flexibility of choice with no implied preference
- 293 **3.1.8**
294 **need not**
295 indicates a course of action permissible within the limits of the document
- 296 **3.1.9**
297 **not recommended**
298 indicates that valid reasons may exist in particular circumstances when the particular behavior is
299 acceptable or even useful, but the full implications should be understood and carefully weighed before
300 implementing any behavior described with this label
- 301 **3.1.10**
302 **obsolete**
303 indicates that an item was defined in prior specifications but has been removed from this specification
- 304 **3.1.11**
305 **optional**
306 indicates a course of action permissible within the limits of the document
- 307 **3.1.12**
308 **recommended**
309 indicates that valid reasons may exist in particular circumstances to ignore a particular item, but the full
310 implications should be understood and carefully weighed before choosing a different course
- 311 **3.1.13**
312 **required**
313 indicates that the item is an absolute requirement of the specification
- 314 **3.1.14**
315 **shall**
316 indicates that the item is an absolute requirement of the specification
317

318 **3.1.15**
319 **shall not**
320 indicates that the definition is an absolute prohibition of the specification

321 **3.1.16**
322 **should**
323 indicates that among several possibilities, one is recommended as particularly suitable, without
324 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required

325 **3.1.17**
326 **should not**
327 indicates that a certain possibility or course of action is deprecated but not prohibited

328 **3.2 MCTP term definitions**

329 For the purposes of this document, the following terms and definitions apply.

330 **3.2.1**
331 **Address Resolution Protocol**
332 **ARP**
333 refers to the procedure used to dynamically determine the addresses of devices on a shared
334 communication medium

335 **3.2.2**
336 **baseline transmission unit**
337 the required common denominator size of a transmission unit for packet payloads that are carried in an
338 MCTP packet. Baseline Transmission Unit-sized packets are guaranteed to be routable within an MCTP
339 network.

340 **3.2.3**
341 **baseboard management controller**
342 **BMC**
343 a term coined by the IPMI specifications for the main management controller in an IPMI-based platform
344 management subsystem. Also sometimes used as a generic name for a motherboard resident
345 management controller that provides motherboard-specific hardware monitoring and control functions for
346 the platform management subsystem.

347 **3.2.4**
348 **binary-coded decimal**
349 **BCD**
350 indicates a particular binary encoding for decimal numbers where each four bits (*nibble*) in a binary
351 number is used to represent a single decimal digit, and with the least significant four bits of the binary
352 number corresponding to the least significant decimal digit. The binary values `0000b` through `1001b`
353 represent decimal values 0 through 9, respectively. For example, with BCD encoding a byte can
354 represent a two-digit decimal number where the most significant nibble (bits 7:4) of the byte contains the
355 encoding for the most significant decimal digit and the least significant nibble (bits 3:0) contains the
356 encoding for the least significant decimal digit (for example, `0010_1001b` in BCD encoding corresponds to
357 the decimal number 29).

- 358 **3.2.5**
359 **bridge**
360 generically, the circuitry and logic that connects one computer bus or interconnect to another, allowing an
361 agent on one to access the other. Within this document, the term *bridge* shall refer to MCTP bridge,
362 unless otherwise indicated.
- 363 **3.2.6**
364 **burst**
365 a number of consecutive baseline transmission unit Packets that the transmitter endpoint sends with
366 minimal delay between those baseline transmission unit packets.
- 367 **3.2.7**
368 **bus**
369 a physical addressing domain shared between one or more platform components that share a common
370 physical layer address space
- 371 **3.2.8**
372 **bus owner**
373 the party responsible for managing address assignments (can be logical or physical addresses) on a bus
374 (for example, in MCTP, the bus owner is the party responsible for managing EID assignments for a given
375 bus). A bus owner may also have additional media-specific responsibilities, such as assignment of
376 physical addresses.
- 377 **3.2.9**
378 **byte**
379 an 8-bit quantity. Also referred to as an *octet*.
380 NOTE: PMCI specifications shall use the term *byte*, not *octet*.
- 381 **3.2.10**
382 **endpoint**
383 see [MCTP endpoint](#)
- 384 **3.2.11**
385 **endpoint ID**
386 **EID**
387 see [MCTP endpoint ID](#)
- 388 **3.2.12**
389 **Globally Unique Identifier**
390 **GUID**
391 see [UUID](#)
- 392 **3.2.13**
393 **host interface**
394 a hardware interface and associated protocols that is used by software running locally on the host
395 processors to access the hardware of a management subsystem within a managed system.

396 **3.2.14**397 **Inter-Integrated Circuit**398 **I²C**

399 a multi-master, two-wire, serial bus originally developed by Philips Semiconductor; now maintained by
400 NXP Semiconductors,

401 **3.2.15**402 **Intelligent Platform Management Bus**403 **IPMB**

404 name for the architecture, protocol, and implementation of an I²C bus that provides a communications
405 path between "management controllers" in IPMI -based systems

406 **3.2.16**407 **Intelligent Platform Management Interface**408 **IPMI**

409 a set of specifications defining interfaces and protocols originally developed for server platform
410 management by the IPMI Promoters Group: Intel, Dell, HP, and NEC

411 **3.2.17**412 **managed entity**

413 the physical or logical entity that is being managed through management parameters. Examples of
414 *physical* entities include fans, processors, power supplies, circuit cards, chassis, and so on. Examples of
415 *logical* entities include virtual processors, cooling domains, system security states, and so on.

416 **3.2.18**417 **Management Component Transport Protocol**418 **MCTP**

419 The protocol defined in this specification.

420 **3.2.19**421 **management controller**

422 a microcontroller or processor that aggregates management parameters from one or more managed
423 devices and makes access to those parameters available to local or remote software, or to other
424 management controllers, through one or more management data models. Management controllers may
425 also interpret and process management-related data, and initiate management-related actions on
426 managed devices. While a native data model is defined for PMCI, it is designed to be capable of
427 supporting other data models, such as CIM, IPMI, and vendor-specific data models. The microcontroller
428 or processor that serves as a management controller can also incorporate the functions of a management
429 device.

430 **3.2.20**431 **managed device**

432 for this specification, managed device refers to a device that is typically implemented using a
433 microcontroller and accessed through a messaging protocol and is used for accessing one or more
434 management parameters. Management parameter access provided by a managed device is typically
435 accomplished using an abstracted interface and data model rather than through direct "register level"
436 accesses. A managed device responds to management requests, but does not initiate or aggregate
437 management operations except in conjunction with a management controller (that is, it is a *satellite*
438 device that is subsidiary to one or more management controllers).

439 **3.2.21**440 **management parameter**

441 a particular datum representing a characteristic, capability, status, or control point associated with a
442 managed entity. Example management parameters include temperature, speed, volts, on/off, link state,
443 uncorrectable error count, device power state, and so on.

444 **3.2.22**445 **MCTP bridge**

446 an MCTP endpoint that can route MCTP messages not destined for itself that it receives on one
447 interconnect onto another without interpreting them. The ingress and egress media at the bridge may be
448 either homogeneous or heterogeneous. Also referred to in this document as a "bridge".

449 **3.2.23**450 **MCTP bus owner**

451 responsible for EID assignment for MCTP or translation on the buses that it is a master of. The MCTP bus
452 owner may also be responsible for physical address assignment. For example, for SMBus/I2C bus
453 segments, the MCTP bus owner is also the ARP master. This means the bus owner assigns dynamic
454 SMBus/I2C addresses to those devices requiring it.

455 **3.2.24**456 **MCTP control command**

457 commands defined under the MCTP *control* message type that are used for the initialization and
458 management of MCTP communications (for example, commands to assign EIDs, discover device MCTP
459 capabilities, and so on)

460 **3.2.25**461 **MCTP endpoint**

462 an MCTP communication terminus. An MCTP endpoint is a terminus or origin of MCTP packets or
463 messages. That is, the combined functionality within a physical device that communicates using the
464 MCTP transport protocol and handles MCTP control commands. This includes MCTP-capable
465 management controllers and managed devices. Also referred to in this document as "endpoint".

466 **3.2.26**467 **MCTP endpoint ID**

468 the logical address used to route MCTP messages to a specific MCTP endpoint. A numeric handle
469 (logical address) that uniquely identifies a particular MCTP endpoint within a system for MCTP
470 communication and message routing purposes. Endpoint IDs are unique among MCTP endpoints that
471 comprise an MCTP communication network within a system. MCTP EIDs are only unique within a
472 particular MCTP network. That is, they can be duplicated or overlap from one MCTP network to the next.
473 Also referred to in this document as "endpoint ID" and abbreviated "EID".

474 **3.2.27**475 **MCTP host interface**

476 a host interface that enables host software to locally access an MCTP Network in the managed system.

477 **3.2.28**478 **MCTP management controller**

479 a management controller that is an MCTP endpoint. Unless otherwise indicated, the term "management
480 controller" refers to an "MCTP management controller" in this document.

- 481 **3.2.29**
482 **MCTP managed device**
483 a managed device that is an MCTP endpoint. Unless otherwise indicated, the term "managed device"
484 refers to an "MCTP managed device" in this document.
- 485 **3.2.30**
486 **MCTP message**
487 a unit of communication based on the message type that is relayed through the MCTP Network using one
488 or more MCTP packets
- 489 **3.2.31**
490 **MCTP network**
491 a collection of MCTP endpoints that communicate using MCTP and share a common MCTP endpoint ID
492 space
- 493 **3.2.32**
494 **MCTP network ID**
495 a unique identifier to distinguish each independent MCTP network within a platform
- 496 **3.2.33**
497 **MCTP packet**
498 the unit of data transfer used for MCTP communication on a given physical medium
- 499 **3.2.34**
500 **MCTP packet payload**
501 refers to the portion of the message body of an MCTP message that is carried in a single MCTP packet
- 502 **3.2.35**
503 **message**
504 see [MCTP message](#)
- 505 **3.2.36**
506 **message assembly**
507 the process of receiving and linking together two or more MCTP packets that belong to a given MCTP
508 message to allow the entire message header and message data (payload) to be extracted
- 509 **3.2.37**
510 **message body**
511 the portion of an MCTP message that carries the message type field and any message type-specific data
512 associated with the message. An MCTP message spans multiple MCTP packets when the message body
513 needs is larger than what can fit in a single MCTP packet. Thus, the message body portion of an MCTP
514 message can span multiple MCTP packets.
- 515 **3.2.38**
516 **message disassembly**
517 the process of taking an MCTP message where the message's header and data (payload) cannot be
518 carried in a single MCTP packet and generating the sequence of two or more packets required to deliver
519 that message content within the MCTP network
- 520 **3.2.39**
521 **message originator**
522 the original transmitter (source) of a message targeted to a particular message terminus

- 523 **3.2.40**
524 **message terminus**
525 the name for a triplet of fields called the MCTP Source Endpoint ID, Tag Owner bit value, and Message
526 Tag value. Together, these fields identify the packets for an MCTP message within an MCTP network for
527 the purpose of message assembly. The message terminus itself can be thought of as identifying a set of
528 resources within the recipient endpoint that is handling the assembly of a particular message.
- 529 **3.2.41**
530 **most significant byte**
531 **MSB**
532 refers to the highest order byte in a number consisting of multiple bytes
- 533 **3.2.42**
534 **nibble**
535 the computer term for a four-bit aggregation, or half of a byte
- 536 **3.2.43**
537 **packet**
538 see [MCTP packet](#)
- 539 **3.2.44**
540 **packet payload**
541 see [MCTP packet payload](#)
- 542 **3.2.45**
543 **pass-through traffic/message/packets**
544 non-control packets passed between the external network and the management controller through the
545 network controller
- 546 **3.2.46**
547 **payload**
548 refers to the information bearing fields of a message. This is separate from those fields and elements that
549 are used to transport the message from one point to another, such as address fields, framing bits,
550 checksums, and so on. In some instances, a given field may be both a payload field and a transport field.
- 551 **3.2.47**
552 **physical transport binding**
553 refers to specifications that define how the MCTP base protocol and MCTP control commands are
554 implemented on a particular physical transport type and medium, such as SMBus/I²C, PCI Express™
555 Vendor Defined Messaging, and so on.
- 556 **3.2.48**
557 **Platform Management Component Intercommunications**
558 **PMCI**
559 name for a working group under the Distributed Management Task Force's Pre-OS Workgroup that is
560 chartered to define standardized communication protocols, low level data models, and transport
561 definitions that support communications with and between management controllers and managed devices
562 that form a platform management subsystem within a managed computer system

563 **3.2.49**

564 **point-to-point**

565 refers to the case where only two physical communication devices are interconnected through a physical
566 communication medium. The devices may be in a master/slave relationship, or could be peers.

567 **3.2.50**

568 **Rate Limiting**

569 a method for limiting the data rate sent from an MCTP endpoint to another MCTP endpoint.

570 **3.2.51**

571 **Reduced Media Independent Interface**

572 **RMII**

573 a reduced signal count MAC to PHY interface, based on the IEEE Media Independent Interface (MII),
574 which was specified by the RMII Consortium (3Com Corporation; AMD Inc.; Bay Networks, Inc.;
575 Broadcom Corp.; National Semiconductor Corp.; and Texas Instruments Inc.)

576 **3.2.52**

577 **simple endpoint**

578 an MCTP endpoint that is not associated with either the functions of an MCTP bus owner or an MCTP
579 bridge

580 **3.2.53**

581 **Transmission Unit**

582 refers to the size of the portion of the MCTP packet payload, which is the portion of the message body
583 carried in an MCTP packet

584 **3.2.54**

585 **transport binding**

586 see [physical transport binding](#)

587 **3.2.55**

588 **Universally Unique Identifier**

589 **UUID**

590 refers to an identifier originally standardized by the Open Software Foundation (OSF) as part of the
591 Distributed Computing Environment (DCE). UUIDs are created using a set of algorithms that enables
592 them to be independently generated by different parties without requiring that the parties coordinate to
593 ensure that generated IDs do not overlap. In this specification, [RFC4122](#) is used as the base specification
594 describing the format and generation of UUIDs. Also sometimes referred to as a globally unique identifier
595 (GUID).

596 **4 Symbols and abbreviated terms**

597 The following symbols and abbreviations are used in this document.

598 **4.1**

599 **ACPI**

600 Advanced Configuration and Power Interface

601 **4.2**

602 **ARP**

603 Address Resolution Protocol

604	4.3
605	BCD
606	binary-coded decimal
607	4.4
608	BMC
609	baseboard management controller
610	4.5
611	CIM
612	Common Information Model
613	4.6
614	EID
615	endpoint identifier
616	4.7
617	FIFO
618	first-in first-out
619	4.8
620	GUID
621	Globally Unique Identifier
622	4.9
623	I²C
624	Inter-Integrated Circuit
625	4.10
626	IANA
627	Internet Assigned Numbers Authority
628	4.11
629	IP
630	Internet Protocol
631	4.12
632	IPMB
633	Intelligent platform management bus
634	4.13
635	IPMI
636	Intelligent platform management interface
637	4.14
638	ISO/IEC
639	International Organization for Standardization/International Engineering Consortium

640	4.15
641	KCS
642	Keyboard Controller Style
643	4.16
644	MCTP
645	Management Component Transport Protocol
646	4.17
647	MSB
648	most significant byte
649	4.18
650	PCIe
651	Peripheral Component Interconnect (PCI) Express
652	4.19
653	PMCI
654	Platform Management Component Intercommunications
655	4.20
656	RMII
657	Reduced Media Independent Interface
658	4.21
659	SMBus
660	System Management Bus
661	4.22
662	TCP/IP
663	Transmission Control Protocol/Internet Protocol
664	4.23
665	USB
666	Universal Serial Bus
667	4.24
668	UUID
669	Universally Unique Identifier
670	4.25
671	VDM
672	Vendor Defined Message

673 **5 Conventions**

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

675 **5.1 Byte ordering**

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

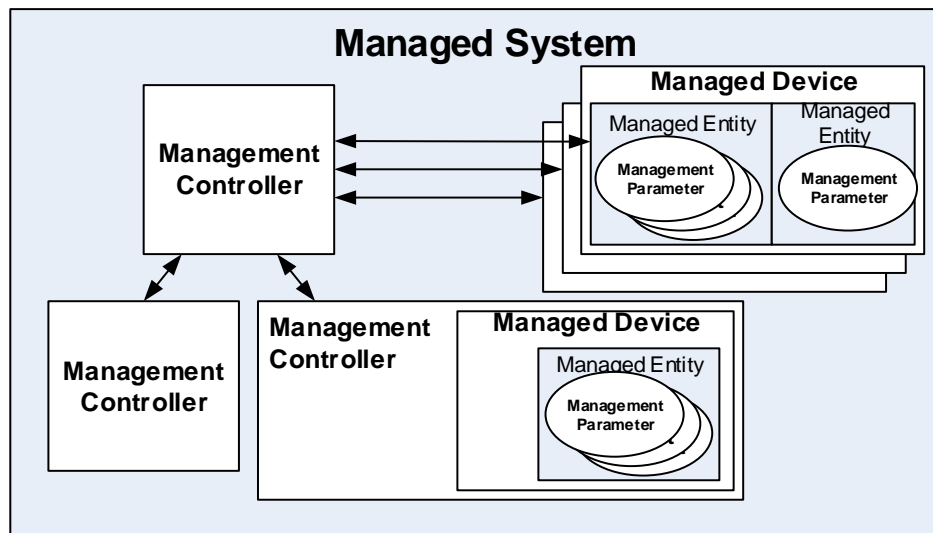
678 **5.2 Reserved fields**

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

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

683 **6 Management component relationships**

684 Figure 1 illustrates the relationship between devices, management controllers, managed devices, and
685 managed entities, which are described in Clause 3.2.



686

687 **Figure 1 – Management component relationships**

688 **7 MCTP overview**

689 This clause provides an overview of the main elements of MCTP. Additional overview information is
690 available in the MCTP white paper, [DSP2016](#).

691 MCTP is a transport independent protocol that is used for intercommunication within an MCTP Network.
692 An MCTP Network that consists of one or more physical transports that are used to transfer MCTP
693 Packets between MCTP Endpoints. MCTP Transport Binding Specifications define how the MCTP
694 protocol is implemented across a particular physical transport medium. For example, the DMTF has

695 defined transport bindings for MCTP over [SMBus/I²C](#) and MCTP over PCIe using PCIe Vendor Defined
 696 Messages (VDMs), and others.

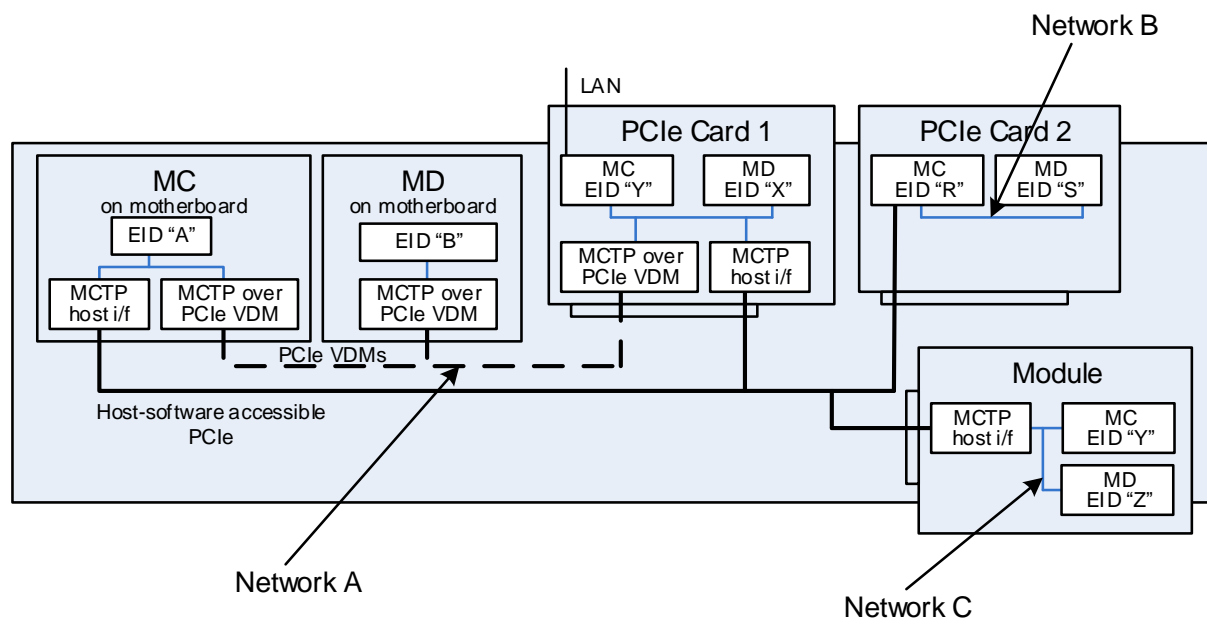
697 An MCTP Endpoint is the terminus for MCTP communication. A physical device that supports MCTP may
 698 provide one or more MCTP Endpoints. Endpoints are addressed using a logical address called the
 699 Endpoint ID, or EID. EIDs in MCTP are analogous to IP Addresses in Internet Protocol networking. EIDs
 700 can be statically or dynamically allocated.

701 A system implementation can contain multiple MCTP Networks. Each MCTP Network has its own
 702 separate EID space. There is no coordination of EIDs between MCTP Networks. EIDs can overlap
 703 between MCTP Networks.

704 An MCTP Network may provide an MCTP Network ID that can be used to differentiate different MCTP
 705 Networks when more than one MCTP Network can be accessed by an entity such as system software.
 706 The Network ID is also used when an entity has more than one point of access to the MCTP Network. In
 707 this case, the MCTP Network ID enables the entity to tell whether the access points provide access to the
 708 same MCTP Network or to different MCTP Networks.

709 The DMTF MCTP specifications also include the definition of transport bindings for MCTP host interfaces.
 710 MCTP host interfaces are used by software that runs locally on the host processors of the managed
 711 system to access an MCTP Network.

712



713

714

Figure 2 – MCTP networks

715 Figure 2 shows the different ways MCTP Networks can exist in a system. In this example, Network A
 716 connects a Management Controllers (MC) and managed devices (MD) on a motherboard with devices on
 717 PCIe Card 1 using MCTP over PCIe Vendor Defined Messages. Note that there are two host interfaces
 718 (host i/f) on standard PCIe (host software accessible) that can be used by host software to access this
 719 particular network. This network thus requires an MCTP Network ID so that the host software can tell that
 720 the two host interfaces connect to the same MCTP Network.

721 Network B represents a network that is solely used for interconnecting devices within PCIe Card 2. This
722 MCTP Network would typically not require an MCTP Network ID since it is not visible to host software or
723 any other entity that would need to differentiate Network B from another MCTP Network in the system.

724 Network C represents an MCTP Network on an add-in module. This network is separate from networks A
725 and B but can be accessed by host software through PCIe. Thus, this network requires a Network ID so that
726 host software can differentiate that Network C is a different network than Network A.

727 MCTP Messages are comprised of one or more MCTP Packets. MCTP defines fields that support the
728 assembly of received MCTP Packets into MCTP Messages and the disassembly of MCTP Messages into
729 packets for transmission.

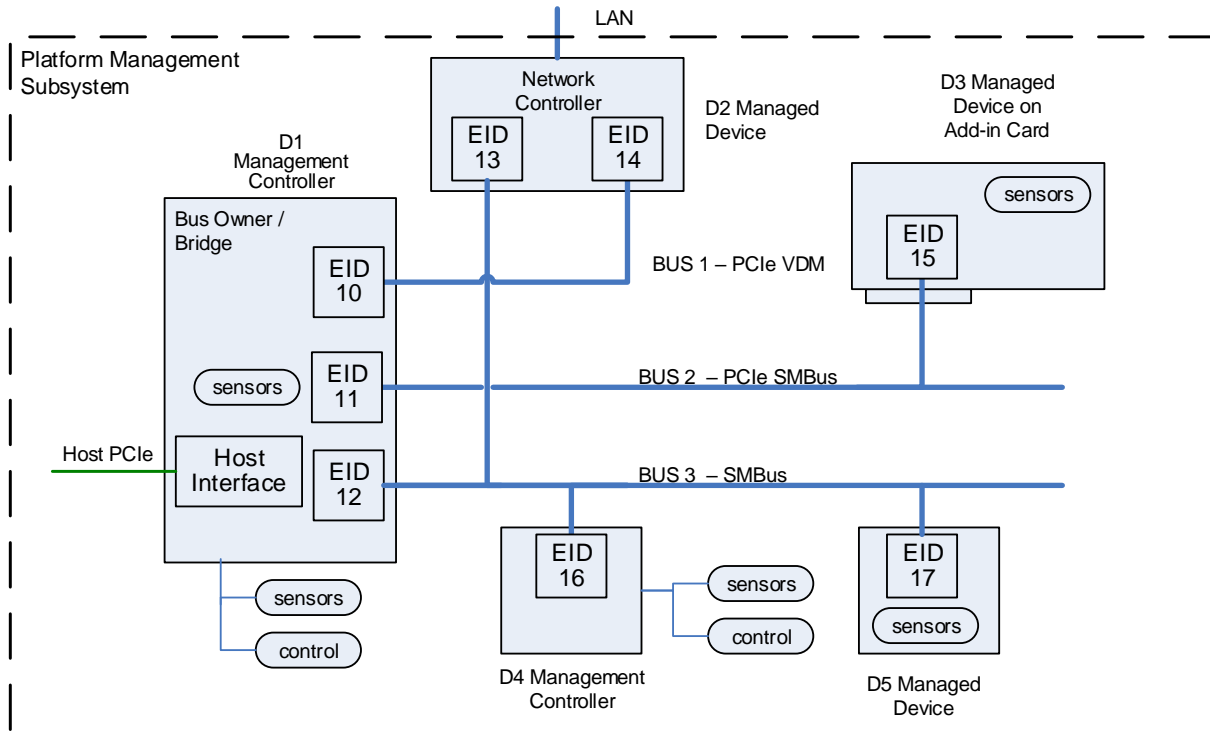
730 MCTP is designed to be able to transfer multiple Message Types in an interleaved manner using the
731 same protocol. MCTP Message Types are identified using a Message Type number. The use of the message
732 type number is similar to a well-known port number in Internet Protocol. It identifies MCTP Messages that
733 are all associated with a particular specification. This specification defines a Message Type for MCTP
734 Control Messages that are used to initialize and maintain the MCTP Network. The DMTF has also defined
735 Message Types for use by the PMCI (Platform Management Communications Interconnect)
736 specifications, Vendor-specific Messaging over MCTP, and so on. MCTP Message Type number
737 assignments are provided in [DSP0239](#). [DSP0239](#) will be updated as new message types are defined in
738 the future.

739 MCTP Control Messages use a request/response protocol. It is important to note that the base transport
740 protocol defined by MCTP just defines a protocol for the transport of MCTP messages. Whether the
741 message content is a request, a response, or something else is part of the particular Message Type
742 definition.

743 In MCTP, a Bus is defined as a physical medium that shares a single physical address space. MCTP
744 includes the definition of a function called the MCTP Bus Owner. The Bus Owner provides two main
745 functions: It distributes EIDs to Endpoints when the MCTP implementation uses EIDs that are dynamically
746 allocated, and it provides the way for an Endpoint to resolve an EID into the physical address used that is
747 required to deliver a message to the target Endpoint.

748 Busses can be interconnected within an MCTP Network using MCTP Bridges to forward MCTP packets
749 between busses. Bridges also handle the task of managing the difference in moving packets from one
750 type of physical media to another, such as moving an MCTP packet between SMBus/I2C and PCIe
751 Vendor Defined Messaging.

752 The following example illustrates how MCTP can be used within a hypothetical platform management
753 subsystem implementation. More complex topologies, with multi-levels of bridges and greater numbers of
754 busses and devices can be readily supported by MCTP as required.



755

756

Figure 3 – MCTP topology

757

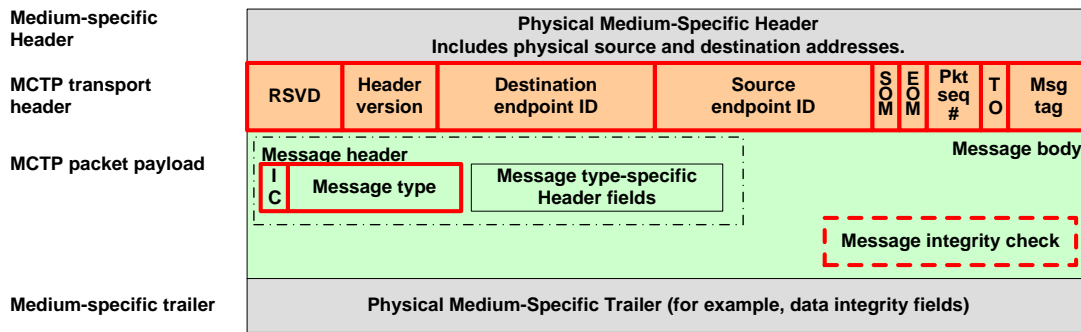
758 **8 MCTP base protocol**

759 The MCTP base protocol defines the common fields for MCTP packets and messages and their usage.

760 Though there are medium-specific packet header fields and trailer fields, the fields for the base protocol
 761 are common for all media. These common fields support the routing and transport of messages between
 762 MCTP endpoints and the assembly and disassembly of large messages from and into multiple MCTP
 763 packets, respectively. The base protocol's common fields include a message type field that identifies what
 764 particular higher layer class of message is being carried using the MCTP base protocol.

765 **8.1 MCTP packet fields**

766 Figure 4 shows the fields that constitute a generic MCTP packet.



767

768 **Figure 4 – Generic message fields**

769 Table 1 defines the base protocol common fields.

770 **Table 1 – MCTP base protocol common fields**

Field Name	Field Size	Description
Medium-specific header	see description	This field represents the physical addressing and framing information that is used for transferring MCTP packets between devices on a particular physical medium. The size and type of any sub-fields or data within this field are defined by the corresponding transport binding specification for MCTP messaging on a given medium (for example, MCTP over SMBus/I2C, MCTP over PCIe Vendor Defined Messaging, and so on).
Medium-specific trailer	see description	This field represents any additional medium-specific trailer fields (if any) that are required for transferring MCTP packets between devices on a particular physical medium. A typical use of this field would be to hold per-packet data integrity fields (for example CRC, checksum, and so on) that would be specified for the particular medium.
MCTP transport header	32 bits	The MCTP transport header is part of each MCTP packet and provides version and addressing information for the packet as well as flags and a "Message Tag" field that, in conjunction with the source EID, is used to identify packets that constitute an MCTP message. The MCTP transport header fields are common fields that are always present regardless of the physical medium over which MCTP is being used. Note: The positioning of the sub-fields of the MCTP transport header may vary based on the physical medium binding.

Field Name	Field Size	Description
RSVD	4 bits	(Reserved) Reserved for future definition by the MCTP base specification.
Hdr version	4 bits	(Header version) Identifies the format, physical framing, and data integrity mechanism used to transfer the MCTP common fields in messages on a given physical medium. The value is defined in the specifications for the particular medium. Note: The value in this field can vary between different transport bindings.
Destination endpoint ID	8 bits	The EID for the endpoint to receive the MCTP packet. A few EID values are reserved for specific routing. See Table 2 – Special endpoint IDs.
Source endpoint ID	8 bits	The EID of the originator of the MCTP packet. See Table 2 – Special endpoint IDs.
SOM	1 bit	(Start Of Message) Set to 1b if this packet is the first packet of a message.
EOM	1 bit	(End Of Message) Set to 1b if this packet is the last packet of a message.
Pkt Seq #	2 bits	(Packet sequence number) For messages that span multiple packets, the packet sequence number increments modulo 4 on each successive packet. This allows the receiver to detect up to three successive missing packets between the start and end of a message. Though the packet sequence number can be any value (0-3) if the SOM bit is set, it is recommended that it is an increment modulo 4 from the prior packet with an EOM bit set. After the SOM packet, the packet sequence number shall increment modulo 4 for each subsequent packet belonging to a given message up through the packet containing the EOM flag.
TO	1 bit	The TO (Tag Owner) bit identifies whether the message tag was originated by the endpoint that is the source of the message or by the endpoint that is the destination of the message. The Message Tag field is generated and tracked independently for each value of the Tag Owner bit. MCTP message types may overlay this bit with additional meaning, for example using it to differentiate between "request" messages and "response" messages. Set to 1b to indicate that the source of the message originated the message tag.
Msg tag	3 bits	(Message tag) Field that, along with the Source Endpoint IDs and the Tag Owner (TO) field, identifies a unique message at the MCTP transport level. Whether other elements, such as portions of the MCTP Message Data field, are also used for uniquely identifying instances or tracking retries of a message is dependent on the message type. A source endpoint is allowed to interleave packets from multiple messages to the same destination endpoint concurrently, provided that each of the messages has a unique message tag. When request/response message exchange is used and the Tag Owner (TO) bit is set to 1 in the request, a responder should return the same Message Tag with the Message Tag Owner bit cleared to 0 in the corresponding response Message. For messages that are split up into multiple packets, the Tag Owner (TO) and Message Tag bits remain the same for all packets from the SOM through the EOM.
Message body	See description	The message body represents the payload of an MCTP message. The message body can span multiple MCTP packets.
IC	1 bit	(MCTP integrity check bit) Indicates whether the MCTP message is covered by an overall MCTP message payload integrity check. This field is required to be the most significant bit of the first byte of the message body in the first packet of a message along with the message type bits. 0b = No MCTP message integrity check 1b = MCTP message integrity check is present

Field Name	Field Size	Description
Message type	7 bits	Defines the type of payload contained in the message data portion of the MCTP message. This field is required to be contained in the least-significant bits of the first byte of the message body in the first packet of a message. Like the fields in the MCTP transport header, the message type field is one of the common MCTP fields that are present independent of the transport over which MCTP is being used. Unlike the MCTP transport header, however, the message type field is only required to be present in the first packet of a particular MCTP message, whereas the MCTP transport header fields are present in every MCTP packet. See DSP0239 and Table 3 for information on message type values.
Message header	0 to M bytes	Additional header information associated with a particular message type, if any. This will typically only be contained in the first packet of a message, but a given message type definition can define header fields as required for any packet.
Message data	0 to N bytes	Data associated with the particular message type. Defined according to the specifications for the message type.
MCTP packet payload	See description	The packet payload is the portion of the message body that is carried in a given MCTP packet. The packet payload is limited according to the rules governing packet payload and transfer unit sizes. See 8.3, Packet payload and transmission unit sizes, for more information.
Msg integrity check	Message type-specific	(MCTP message integrity check) This field represents the optional presence of a message type-specific integrity check over the contents of the message body. If present, the Message integrity check field shall be carried in the last bytes of the message body. The particular message type definition will specify whether this is required, optional, or not to be used, the field size, and what algorithm is to be used to generate the field. The MCTP base protocol also does not specify whether this field is required on single packet messages (potentially dependent on transmission unit size) or is only required on multiple packet messages. Use of the Msg integrity check field is specific to the particular message type specification.

771 **8.2 Special endpoint IDs**

772 The following table lists EID values that are reserved or assigned to specific functions for MCTP.

773 **Table 2 – 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.

Value	Description
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.

774 8.3 Packet payload and transmission unit sizes

775 For MCTP, the size of a transmission unit is defined as the size of the packet payload that is carried in an
776 MCTP packet.

777 8.3.1 Baseline transmission unit

778 The following are key information points regarding baseline transmission unit:

- 779 • The baseline transmission unit (minimum transmission unit) size for MCTP is 64 bytes.
- 780 • A message terminus that supports MCTP control messages shall always accept valid packets
781 that have a transmission unit equal to or less than the baseline transmission unit. The message
782 terminus is also allowed to support larger transmission units.
- 783 • The transmission unit of all packets in a given message shall be the same size, except for the
784 transmission unit in the last packet (packet with EOM bit = 1b). Except for the last packet, this
785 size shall be at least the baseline transmission unit size.
- 786 • The size of the transmission unit in the last packet shall be less than or equal to the
787 transmission unit size used for the other packets (if any).
- 788 • If a transmission unit size larger than the baseline transmission unit is negotiated, the
789 transmission unit of all packets shall be less than or equal to the negotiated transmission unit
790 size. (The negotiation mechanism for larger transmission units between endpoints is message
791 type-specific and is not addressed in this specification.)
- 792 • A given endpoint may negotiate additional restrictions on packet sizes for communication with
793 another endpoint, as long as the requirements of this clause are met.
- 794 • All message types shall include support for being delivered using packets that have a
795 transmission unit that is no larger than the baseline transmission unit. This is required to support
796 bridging those messages in implementations where there are MCTP bridges that only support
797 the baseline transmission unit.

798 8.4 Maximum message body sizes

799 The Message Body can span multiple packets. Limitations on message body sizes are message type-
800 specific and are documented in the specifications for each message type.

801 8.5 Message assembly

802 The following fields (and *only* these fields) are collectively used to identify the packets that belong to a
803 given message for the purpose of message assembly on a particular destination endpoint.

- 804 • Msg Tag (Message Tag)
- 805 • TO (Tag Owner)
- 806 • Source Endpoint ID

807 As described in 3.2, together these values identify the message terminus on the destination endpoint. For
808 a given message terminus, only one message assembly is allowed to be in process at a time.

8.6 Dropped packets

Individual packets are dropped (silently discarded) by an endpoint under the following conditions. These packets are discarded before being checked for acceptance or rejection for message assembly. Therefore, these packets will *not* cause a message assembly to be started or terminated.

- **Unexpected "middle" packet or "end" packet**

A "middle" packet (SOM flag = 0 and EOM flag = 0) or "end" packet (SOM flag = 0 and EOM flag = 1) for a multiple-packet message is received for a given message terminus without first having received a corresponding "start" packet (where the "start" packet has SOM flag = 1 and EOM flag = 0) for the message.

- **Bad packet data integrity or other physical layer error**

A packet is dropped at the physical data-link layer because a data integrity check on the packet at that layer was invalid. Other possible physical layer errors may include framing errors, byte alignment errors, packet sizes that do not meet the physical layer requirements, and so on.

- **Bad, unexpected, or expired message tag**

A message with TO bit = 0 was received, indicating that the destination endpoint was the originator of the tag value, but the destination endpoint did not originate that value, or is no longer expecting it. (MCTP bridges do not check message tag or TO bit values for messages that are not addressed to the bridge's EID, or to the bridge's physical address if null-source or destination-EID physical addressing is used.)

- **Unknown destination EID**

A packet is received at the physical address of the device, but the destination EID does not match the EID for the device or the EID is un-routable.

- **Un-routable EID**

An MCTP bridge receives an EID that the bridge is not able to route (for example, because the bridge did not have a routing table entry for the given endpoint).

- **Bad header version**

The MCTP header version (Hdr Version) value is not a value that the endpoint supports.

- **Unsupported transmission unit**

The transmission unit size is not supported by the endpoint that is receiving the packet.

8.7 Starting message assembly

Multiple-packet message assembly begins when the endpoint corresponding to the destination EID in the packet receives a valid "start" packet (packet with SOM = 1b and EOM = 0b).

A packet with both SOM = 1b and EOM = 1b is considered to be a single-packet message, and is not assembled per se.

Both multiple- and single-packet messages are subject to being terminated or dropped based on conditions listed in the following clause.

8.8 Terminating message assembly/dropped messages

Message assembly is terminated at the destination endpoint and messages are accepted or dropped under the following conditions:

- 848 • **Receipt of the "end" packet for the given message**
- 849 Receiving an "end" packet (packet with EOM = 1b) for a message that is in the process of being
850 assembled on a given message terminus will cause the message assembly to be completed
851 (provided that the message has not been terminated for any of the reasons listed below). This is
852 normal termination. The message is considered to be accepted at the MCTP base protocol
853 level.
- 854 • **Receipt of a new "start" packet**
- 855 Receiving a new "start" packet (packet with SOM = 1b) for a message to the same message
856 terminus as a message assembly already in progress will cause the message assembly in
857 process to be terminated. All data for the message assembly that was in progress is dropped.
858 The newly received start packet is not dropped, but instead it begins a new message assembly.
859 This is considered an error condition.
- 860 • **Timeout waiting for a packet**
- 861 Too much time occurred between packets of a given multiple-packet message. All data for the
862 message assembly that was in progress are dropped. This is considered an error condition. The
863 timeout interval, if specified, is specific to the transport binding specification. (A binding
864 specification may choose to not define a value for this timeout.)
- 865 • **Out-of-sequence packet sequence number**
- 866 For packets comprising a given multiple-packet message, the packet sequence number for the
867 most recently received packet is not a mod 4 increment of the previously received packet's
868 sequence number. All data for the message assembly that was in progress is dropped. This is
869 considered an error condition.
- 870 • **Incorrect transmission unit**
- 871 An implementation may terminate message assembly if it receives a "middle" packet (SOM =
872 0b and EOM = 0b) where the MCTP packet payload size does not match the MCTP packet
873 payload size for the start packet (SOM = 1b and EOM bit = 0b). This is considered an error
874 condition.
- 875 • **Bad message integrity check**
- 876 For single- or multiple-packet messages that use a message integrity check, a mismatch with
877 the message integrity check value can cause the message assembly to be terminated and the
878 entire message to be dropped, unless it is overridden by the specification for a particular
879 message type.
- 880 NOTE: The message integrity check is considered to be at the message-type level error condition rather
881 than an error at the MCTP base protocol level.

882 **8.9 Dropped messages**

883 An endpoint may drop a message if the message type is not supported by the endpoint. This can happen
884 in any one of the following ways:

- 885 • The endpoint can elect to not start message assembly upon detecting the invalid message type
886 in the first packet.
- 887 • The endpoint can elect to terminate message assembly in process.
- 888 • The endpoint can elect to drop the message after it has been assembled.

889 **8.10 MCTP versioning and message type support**

890 There are three types of versioning information that can be retrieved using MCTP control messages:

- 891 • MCTP base specification version information
- 892 • MCTP packet header version information
- 893 • Message type version information

894 The version of the MCTP base specification that is supported by a given endpoint is obtained through the
895 Get MCTP Version Support command. This command can also be used to discover whether a particular
896 message type is supported on an endpoint, and if so, what versions of that message type are supported.

897 The Header Version field in MCTP packets identifies the media-specific formatting used for MCTP
898 packets. It can also indicate a level of current and backward compatibility with versions of the base
899 specification, as specified by the header version definition in each medium-specific transport binding
900 specification.

901 **8.10.1 Compatibility with future versions of MCTP**

902 An Endpoint may choose to support only certain versions of MCTP. The command structure along with
903 the Get MCTP Version Support command allows endpoints to detect and restrict the versions of MCTP
904 used by other communication endpoints. To support this, all endpoints on a given medium are required to
905 implement MCTP Version 1.0.x control commands or later 1.x Version for initialization and version
906 support discovery.

907

908 8.11 MCTP message types

909 Table 3 defines the values for the Message Type field for different message types transported through
 910 MCTP. The MCTP control message type is specified within this document. Baseline requirements for the
 911 Vendor Defined – PCI and Vendor Defined – IANA message types are also specified within this
 912 document. All other message types are specified in the [DSP0239](#) companion document to this
 913 specification.

914 NOTE: A device that supports a given message type may not support that message type equally across all buses
 915 that connect to the device.

916 **Table 3 – MCTP Message Types Used in this Specification**

Message Type	Message Type Code	Description
MCTP control	0x00	Messages used to support initialization and configuration of MCTP communication within an MCTP network. The messages and functions for this message type are defined within this specification.
Vendor Defined – PCI	0x7E	Message type used to support VDMs where the vendor is identified using a PCI-based vendor ID. The specification of the initial message header bytes for this message type is provided within this specification. 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 using an IANA-based vendor ID. (This format uses an "enterprise number" that is assigned and maintained by the Internet Assigned Numbers Authority, www.iana.org , as the means of identifying a particular vendor, company, or organization.) The specification of the initial message header bytes for this message type is provided within this specification. Otherwise, the message body content is specified by the vendor, company, or organization identified by the given vendor ID.

917 8.12 Security

918 The basic premise of MCTP is that higher layer protocols will fulfill security requirements (for example,
 919 confidentiality and authentication) for communication of management data. This means that the data
 920 models carried by MCTP shall fulfill the security requirements of a given management transaction. The
 921 MCTP protocol itself will not define any additional security mechanisms.

922 8.13 Limitations

923 MCTP has been optimized for communications that occur within a single computer system platform. It has
 924 not been designed to handle problems that can typically occur in a more generic inter-system networking
 925 environment. In particular, compared to networking protocols such as IP and TCP/IP, MCTP has the
 926 following limitations:

- 927 • MCTP has limited logical addressing. MCTP been optimized for the small number of endpoints
 928 that are expected to be utilized within the platform. The 8-bit range of EIDs is limited compared
 929 to the ranges available for IP addresses.
- 930 • MCTP assumes an MCTP network implementation that does not include loops. There is no
 931 mechanism defined in MCTP to detect or reconcile implementations that have connections that
 932 form routing loops.

- 933 • MCTP assumes a network topology where all packets belonging to a given message will be
934 delivered through the same route (that is, MCTP does not generally support some packets for a
935 message arriving by one route, while other packets for the message arrive by a different route).
 - 936 • MCTP does not support out-of-order packets for message assembly.
 - 937 • The MCTP base protocol does not address flow control or congestion control. These behaviors,
938 if required, are specified at the physical transport binding level or at the message type or higher
939 level.
 - 940 • MCTP is not specified to handle duplicate packets at the base protocol message assembly
941 level. If a duplicate packet is received and passed on to MCTP message assembly, it can cause
942 the entire message assembly to be terminated.
- 943 NOTE: Transport bindings are not precluded from including mechanisms for handling duplicate packets
944 at the physical transport level.

945 **8.14 MCTP discovery and addressing**

946 This clause describes how MCTP endpoints and their capabilities are discovered by one another, and
947 how MCTP endpoints are provisioned with the addresses necessary for MCTP communication.

948 MCTP discovery occurs over the course of several discrete, ordered steps:

- 949 • Bus enumeration
- 950 • Bus address assignment
- 951 • MCTP capability discovery
- 952 • Endpoint ID assignment
- 953 • Distribution and use of routing information

954 This clause gives an overview of the methods used for accomplishing each of these steps in various
955 operational scenarios. Clause 12 gives details on the messages used to implement these operations.

956 **8.14.1 Bus enumeration**

957 This step represents existing bus enumeration. (The actions taken in this step are specific to a given
958 medium.) Because enumeration of devices on the physical bus is medium-specific, this information is
959 provided in the transport binding specification for the medium.

960 **8.14.2 Bus address assignment**

961 MCTP endpoints require a bus address that is unique to a given bus segment. This step deals with
962 assignment of these addresses. Some bus types (such as PCIe) have built-in mechanisms to effectively
963 deal with this. Others (such as SMBus/I2C) require some additional consideration. Because bus address
964 assignment is medium-specific, this information is provided in the transport binding specification for the
965 medium.

966 **8.14.3 MCTP capability discovery**

967 Capability discovery deals with the discovery of the characteristics of individual MCTP endpoints.
968 Capabilities that can be discovered include what message types are supported by an endpoint and what
969 message type versions are supported. See 8.10 for a description of the methods used to accomplish
970 capability discovery.

971 **8.14.4 Endpoint ID assignment**

972 Endpoint IDs are system-wide unique IDs for identifying a specific MCTP endpoint. They can be
973 dynamically assigned at system startup or hot-plug insertion. See 8.17 for a description of the methods
974 used to accomplish EID assignment.

975 **8.14.5 Distribution and use of routing information**

976 Bridging-capable MCTP endpoints need routing information to identify the next hop to forward a message
977 to its final destination. See 9 for a description of how routing information is conveyed between MCTP
978 endpoints.

979 **8.15 Devices with multiple media interfaces**

980 MCTP fully supports management controllers or managed devices that have interfaces on more than one
981 type of bus. For example, a device could have both a PCI Express (PCIe) and an SMBus/I2C interface. In
982 this scenario, the device will typically have a different EID for each interface. (Bridges can include
983 instantiations that have an endpoint shared across multiple interfaces; see 9.1.2 for more information.)

984 This concept can be useful in different operational scenarios of the managed system. For example,
985 typically a PCIe interface will be used during [ACPI](#) "S0" power states (when the system is fully powered
986 up), which will provide significantly higher bandwidths, whereas the SMBus/I2C interface could be used
987 for "S3–S5" low-power sleep states.

988 The baseline transmission unit is specified to be common across all media, enabling packets to be routed
989 between different media without requiring bridges to do intermediate assembly and disassembly
990 operations to handle differences in packet payload sizes between different media.

991 Devices that support multiple media interfaces shall meet the command requirements of this specification
992 and the associated transport binding specification for each enabled interface. For a given message type,
993 the device may implement the same message type –specific commands on all MCTP interfaces,
994 regardless of the medium, unless otherwise specified by the message type specification.

995 **8.16 Peer transactions**

996 Endpoints can intercommunicate in a peer-to-peer manner using the physical addressing on a given bus.

997 A special value for the EID is used in cases when the physical address is known, but the EID is not
998 known. This capability is used primarily to support device discovery and EID assignment. A device that
999 does not yet have an EID assignment is not addressed using an EID. Rather, the device gets its EID
1000 assigned using an MCTP control command, Set Endpoint ID, which uses physical addressing only.

1001 Similarly, depending on the transport binding, a device can also announce its presence by sending an
1002 MCTP message to a well-known physical address for the bus owner (for example, for PCIe VDM, this
1003 would be the root complex; for SMBus/I2C, the host slave address, and so on).

1004 It is important to note that in cases where two endpoints are on the same bus, they do not need to go
1005 through a bridge to communicate with each other. Devices use the Resolve Endpoint ID command to ask
1006 the bus owner what physical address should be used to route messages to a given EID. Depending on
1007 the bus implementation, the bus owner can either return the physical address of the bridge that the
1008 message should be delivered to, or it can return the physical address of the peer on the bus.

1009 **8.17 Endpoint ID assignment and endpoint ID pools**

1010 MCTP EIDs are the system-wide unique IDs used by the MCTP infrastructure to address endpoints and
1011 for routing messages across multiple buses in the system. There is one EID assigned to a given physical
1012 address. Most managed devices or management controllers will connect to just a single bus and have a

1013 single EID. A non-bridge device that is connected to multiple different buses will have one EID for each
 1014 bus it is attached to.

1015 Bus owners are MCTP devices that are responsible for issuing EIDs to devices on a bus segment. These
 1016 EIDs come from a pool of EIDs maintained by the bus owner.

1017 With the exception of the topmost bus owner (see 8.17.1), a given bus owner's pool of EIDs is
 1018 dynamically allocated at run-time by the bus owner of the bus above it in the hierarchy. Hot-plug devices
 1019 shall have their EID pools dynamically allocated.

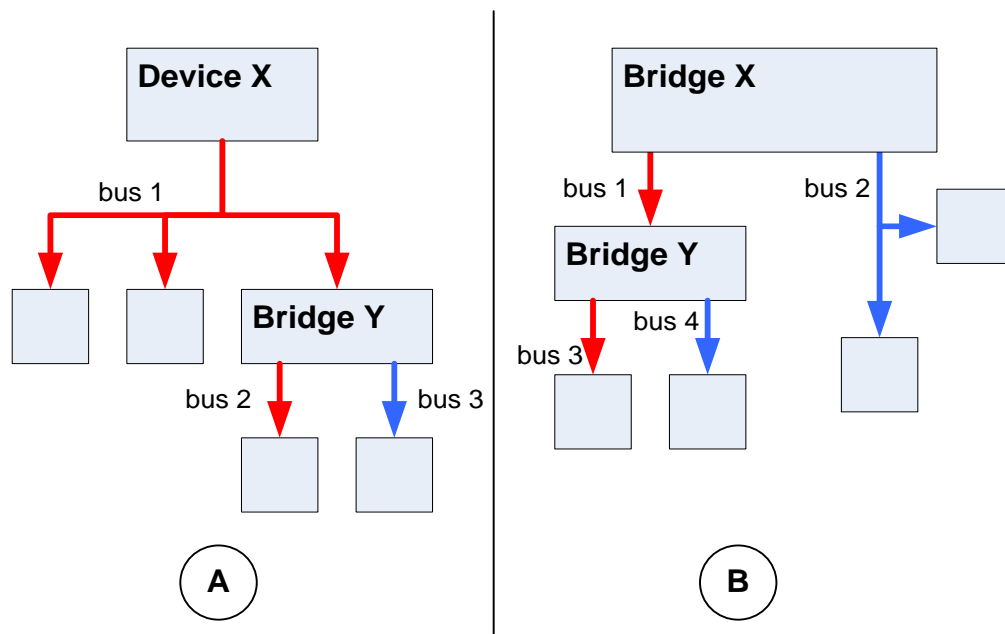
1020 Once EIDs are assigned to MCTP endpoints, it is necessary for MCTP devices involved in a transaction
 1021 to understand something about the route a given message will traverse. Clause 9 describes how this
 1022 routing information is shared among participants along a message's route.

1023 **8.17.1 Topmost bus owner**

1024 The topmost bus owner is the ultimate source of the EID pool from which all EIDs are drawn for a given
 1025 MCTP network.

- 1026 1. This is illustrated in Figure 5, in which the arrows are used to identify the role of bus ownership. The
 1027 arrows point outward from the bus owner for the particular bus and inward to a device that is "owned"
 1028 on the bus.

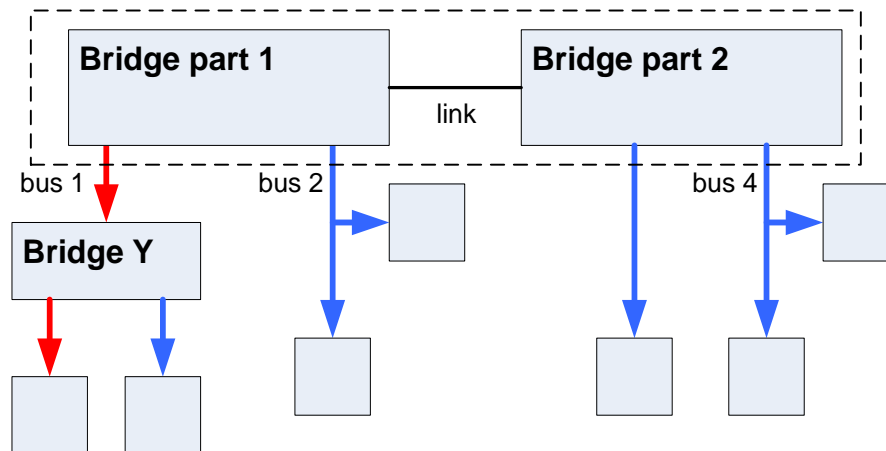
1029 In Figure 5, device X in diagram A and bridge X in diagram B are examples of topmost bus owners.
 1030 Diagram A shows a device that connects to a single bus and is the topmost bus owner for the overall
 1031 MCTP network. Diagram B shows that a bridge can simultaneously be the topmost bus owner, as well as
 1032 the bus owner for more than one bus. The different colors represent examples of different media.



1033

1034

Figure 5 – Topmost bus owners



1035

1036

Figure 6 – Split bridge

1037 An implementation may need to split a bus owner or bridge across two physical devices. Such an
 1038 implementation shall include a mechanism (for example, a link as shown in Figure 6) that enables the two
 1039 parts to share a common routing table, or have individual copies of the routing table that are kept
 1040 synchronized. The definition of this mechanism is outside the scope of this specification.

1041 8.17.2 Use of static EIDs and static EID pools

1042 In general, the only device that will require a static (pre-configured default assigned non-zero value) EID
 1043 assignment will be the topmost bus owner. It needs a static EID because there is no other party to assign
 1044 it an EID through MCTP. Otherwise, all other devices will have their EIDs assigned to them by a bus
 1045 owner.

1046 The same principle applies if the device functions as an MCTP bridge. If the device is the highest device
 1047 in the MCTP bus hierarchy, it will require a static pool of EIDs to be assigned to it as part of the system
 1048 design. Otherwise, the device will be dynamically allocated a pool of EIDs from a higher bus owner.

1049 An MCTP network implementation is allowed to use static EIDs for devices other than the topmost bus
 1050 owner. Typically, this would only be done for very simple MCTP networks. Other key EID assignment
 1051 considerations follow:

- 1052 • Endpoints that support the option of being configured for one or more static EIDs shall also
 1053 support being configured to be dynamically assigned EIDs.
- 1054 • No mechanism is defined in the MCTP base specification for a bridge or bus owner to discover
 1055 and incorporate a static EID into its routing information. Thus, a simple endpoint that is
 1056 configured with a static EID shall also be used with a bus owner that is configured to support the
 1057 static EIDs for the endpoint.
- 1058 • All bus owners/bridges in the hierarchy, from the topmost bus owner to the endpoint, shall have
 1059 their routing configurable to support static EID routing information.
- 1060 • Although an endpoint that uses a static EID shall be used with a bus owner that supports static
 1061 EIDs, the reverse is not true. A bus owner that uses static EIDs does not need to require that
 1062 the devices on the buses it owns be configured with static EIDs.
- 1063 • How the configuration of static EIDs default value occurs is outside the scope of this
 1064 specification.

- 1065 • No specified mechanism exists to "force" an override of a bridge's or bus owner's routing table
1066 entries for static EIDs. That is, commands such as Allocate Endpoint IDs and Routing
1067 Information Update only affect entries that are associated with dynamic EIDs.
- 1068 • MCTP does not define a mechanism for keeping routing tables updated if static EIDs are used
1069 with dynamic physical addresses. That is, static EIDs are not supported for use with dynamic
1070 physical addresses.
- 1071 • Bridges can have a mix of both static and dynamic EID pools. That is, the routing table can
1072 have both static and dynamic entries and can allocate from static and dynamic EID pools. Only
1073 the dynamic EID pool is given to the bridge by the bus owner using the Allocate Endpoint IDs
1074 command. There is no specification for how a static EID pool gets configured or how a bridge
1075 decides whether to give an endpoint an EID from a static or dynamically obtained EID pool.
1076 There is also no MCTP-defined mechanism to read the static EID pool setting from the bridge.
- 1077 • MCTP bridges and bus owners (except the topmost bus owner) are not required to include
1078 support for static EIDs.
- 1079 • MCTP does not define a mechanism for allocating EID pools that take static EID assignments
1080 into account. That is, a bridge cannot request a particular set of EIDs to be allocated to it.
- 1081 • MCTP bridges/bus owners may be configurable to use only static EIDs.

1082 8.17.3 Use of static physical addresses

1083 In many simple topologies, it is desirable to use devices that have statically configured physical
1084 addresses. This can simplify the implementation of the device. For example, an SMBus/I2C device that is
1085 not used in a hot-plug application would not need to support the SMBus address assignment (SMBus
1086 ARP) protocol. Fixed addresses can also aid in identifying the location and use of an MCTP device in a
1087 system. For example, if a system has two otherwise identical MCTP devices, a system vendor will know
1088 that the device at address "X" is the one at the front of the motherboard, and the device at address "Y" is
1089 at the back, because that is how they assigned the addresses when the system was designed.

1090 Therefore, MCTP transport bindings, such as for SMBus/I2C, are allowed to support devices being at
1091 static physical addresses without requiring the binding to define a mechanism that enables the bus owner
1092 to discover MCTP devices that are using static addresses.

1093 In this case, the bridge or bus owner shall have a-priori knowledge of the addresses of those devices to
1094 be able to assign EIDs to those devices and to support routing services for those devices. To support this
1095 requirement, the following requirements and recommendations are given to device vendors:

- 1096 • Devices that act as bus owners or bridges and are intended to support MCTP devices that use
1097 static physical addresses should provide a non-volatile configuration option that enables the
1098 system integrator to configure which device addresses are being used for devices on each bus
1099 that is owned by the bridge/bus owner.
- 1100 • The mechanism by which this non-volatile configuration occurs is specific to the device vendor.
1101 In many cases, the physical address information will be kept in some type of non-volatile
1102 storage that is associated with the device and gets loaded when the device is manufactured or
1103 when the device is integrated into a system. In other cases, this information may be coded into
1104 a firmware build for the device.

1105 **8.17.4 Endpoint ID assignment process for bus owners/bridges**

1106 The bus owner/bridge shall get its own EID assignment, and a pool of EIDs, as follows. These steps only
1107 apply to bus owner/bridge devices that are not the topmost bus owner.

- 1108 • Bus owners/bridges shall be pre-configured with non-volatile information that identifies which
1109 buses they own. (How this configuration is accomplished is device/vendor specific and is
1110 outside the scope of this specification.)
- 1111 • The bus owner/bridge announces its presence on any buses *that it does not own* to get an EID
1112 assignment for that bus. The mechanism by which this announcement occurs is dependent on
1113 the particular physical transport binding and is defined as part of the binding specification.
- 1114 • The bus owner/bridge waits until it gets its own EID assignment for one of those buses through
1115 the Set Endpoint ID command.
- 1116 • The bus owner/bridge indicates the size of the EID pool it requires by returning that information
1117 in the response to the Set Endpoint ID command.
- 1118 • For each bus where the bus owner/bridge is itself an "owned" device, the bus owner/bridge will
1119 be offered a pool of EIDs by being sent an Allocate Endpoint IDs command from the bus owner.
- 1120 • The bus owner/bridge accepts allocations only from the bus of the "first" bus owner that gives it
1121 the allocation, as described in the Allocate Endpoint IDs command description in 8.10. If it gets
1122 allocations from other buses, they are rejected.

1123 The bus owner can now begin to build a routing table for each of the buses that it owns, and accept
1124 routing information update information. Refer to 9 for more information.

1125 **8.17.5 Endpoint ID retention**

1126 Devices should retain their EID assignments for as long as they are in their normal operating state.
1127 Asynchronous conditions, such as device errors, unexpected power loss, power state changes, resets,
1128 firmware updates, may cause a device to require a reassignment of its EID. Devices should retain their
1129 EID assignments across conditions where they may temporarily stop responding to commands over
1130 MCTP, such as during internal resets, error conditions, or configuration updates.

1131 **8.17.6 Reclaiming EIDs from hot-plug devices**

1132 Bridges will typically have a limited pool of EIDs from which to assign and allocate to devices. (This also
1133 applies when a single bus owner supports hot-plug devices.) It is important for bridges to reclaim EIDs so
1134 that when a device is removed, the EID can later be re-assigned when a device is plugged in. Otherwise,
1135 the EID pool could become depleted as devices are successively removed and added.

1136 EIDs for endpoints that use static addresses are not reclaimed.

1137 No mechanism is specified in the MCTP base protocol for detecting device removal when it occurs.
1138 Therefore, the general approach to detecting whether a device has been removed is to re-enumerate the
1139 bus when a new device is added and an EID or EID pool is being assigned to that device.

1140 The following approach can be used to detect removed hot-plug devices: The bus owner/bridge can
1141 detect a removed device or devices by validating the EIDs that are presently allocated to endpoints that
1142 are directly on the bus and identifying which EIDs are missing. It can do this by attempting to access each
1143 endpoint that the bridge has listed in its routing table as being a device that is directly on the particular
1144 bus. Attempting to access each endpoint can be accomplished by issuing the Get Endpoint ID command
1145 to the physical address of each device and comparing the returned result to the existing entry in the
1146 routing table. If there is no response to the command, or if there is a mismatch with the existing routing
1147 information, the entry should be cleared and the corresponding EID or EID range should be returned to
1148 the "pool" for re-assignment. The bus owner/bridge can then go through the normal steps for EID
1149 assignment.

1150 This approach should work for all physical transport bindings, because it keeps the "removed EID"
1151 detection processing separated from the address assignment process for the bus.

1152 In some cases, a hot-plug endpoint may temporarily go into a state where it does not respond to MCTP
1153 control messages. Depending on the medium, it is possible that when the endpoint comes back on line, it
1154 does not request a new EID assignment but instead continues using the EID it had originally assigned. If
1155 this occurs while the bus owner is validating EIDs to see if any endpoints are no longer accessible, it is
1156 possible that the bus owner will assume that the endpoint was removed and reassign its EID to a newly
1157 inserted endpoint, unless other steps are taken:

- 1158 • The bus owner shall wait at least $T_{RECLAIM}$ seconds before reassigning a given EID (where
1159 $T_{RECLAIM}$ is specified in the physical transport binding specification for the medium used to
1160 access the endpoint).
- 1161 • Reclaimed EIDs shall only be reassigned after all unused EIDs in the EID pool have been
1162 assigned to endpoints. Optionally, additional robustness can be achieved if the bus owner
1163 maintains a short FIFO list of reclaimed EIDs (and their associated physical addresses) and
1164 allocates the older EIDs first.
- 1165 • A bus owner shall confirm that an endpoint has been removed by attempting to access it after
1166 $T_{RECLAIM}$ has expired. It can do this by issuing a Get Endpoint ID command to the endpoint to
1167 verify that the endpoint is still non-responsive. It is recommended that this be done at least three
1168 times, with a delay of at least $1/2 * T_{RECLAIM}$ between tries if possible. If the endpoint continues
1169 to be non-responsive, it can be assumed that it is safe to return its EID to the pool of EIDs
1170 available for assignment.

1171 **8.17.7 Additional requirements for hot-plug endpoints**

1172 Devices that are hot-plug shall support the Get Endpoint UUID command. The purpose of this
1173 requirement is to provide a common mechanism for identifying when devices have been changed.

1174 Endpoints that go into states where they temporarily do not respond to MCTP control messages shall re-
1175 announce themselves and request a new EID assignment if they are "off line" for more than $T_{RECLAIM}$
1176 seconds, where $T_{RECLAIM}$ is specified in the physical transport binding specification for the medium used
1177 to access the endpoint.

1178 **8.17.8 Additional requirements for devices with multiple endpoints**

1179 A separate EID is utilized for each MCTP bus that a non-bridge device connects to. In many cases, it is
1180 desirable to be able to identify that the same device is accessible through multiple EIDs.

1181 If an endpoint has multiple physical interfaces (ports), the interfaces can be correlated to the device by
1182 using the MCTP Get Endpoint UUID command (see 12.5) to retrieve the unique system-wide identifier.

1183 Devices connected to multiple buses shall support the Get Endpoint UUID command for each endpoint
1184 and return a common UUID value across all the endpoints. This is to enable identifying EIDs as belonging
1185 to the same physical device.

1186 **8.18 Handling reassigned EIDs**

1187 Though unlikely, it is still possible that during the course of operation of an MCTP network, a particular
1188 EID could get reassigned from one endpoint to another. For example, this could occur if a newly hot-swap
1189 inserted endpoint device gets assigned an EID that was previously assigned to a device that was
1190 subsequently removed.

1191 Under this condition, it is possible that the endpoint could receive a message that was intended for the
1192 previously installed device. This is not considered an issue for MCTP control messages because the
1193 control messages are typically just used by bus owners and bridges for initializing and maintaining the

1194 MCTP network. The bus owners and bridges are aware of the EIDs they have assigned to endpoints and
1195 are thus intrinsically aware of any EID reassignment.

1196 Other endpoints, however, are not explicitly notified of the reassignment of EIDs. Therefore,
1197 communication that occurs directly from one endpoint to another is subject to the possibility that the EID
1198 could become assigned to a different device in the middle of communication. This shall be protected
1199 against by protocols specific to the message type being used for the communication.

1200 In general, the approach to protecting against this will be that other message types will require some kind
1201 of "session" to be established between the intercommunicating endpoints. By default, devices would not
1202 start up with an active session. Thus, if a new device is added and it gets a reassigned EID, it will not
1203 have an active session with the other device and the other device will detect this when it tries to
1204 communicate.

1205 The act of having a new EID assigned to an existing device should have the same effect. That is, if a
1206 device gets a new EID assignment, it would "close" any active sessions for other message types.

1207 The mechanism by which other message types would establish and track communication sessions
1208 between devices is not specified in this document. It is up to the specification of the particular message
1209 type.

1210 **9 MCTP bridging**

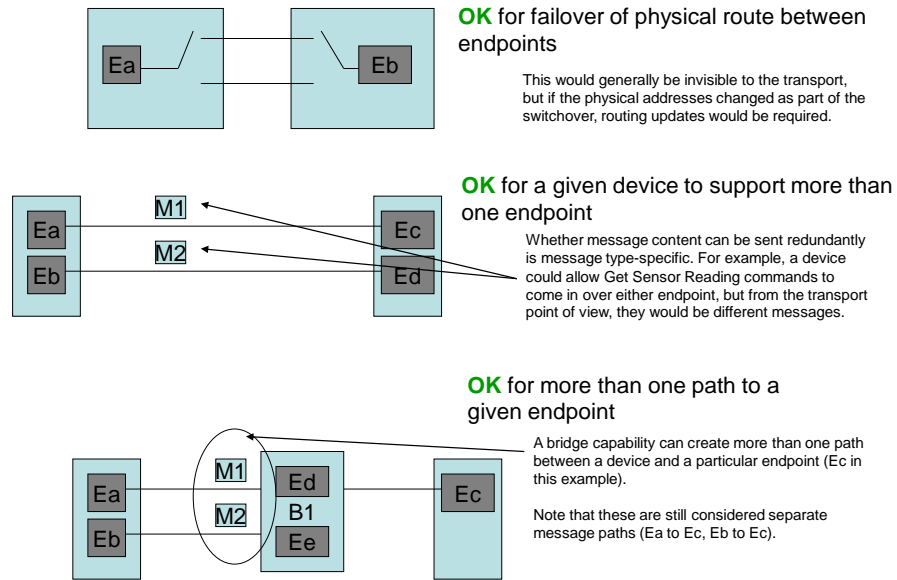
1211 One key capability provided by MCTP is its ability to route messages between multiple buses and
1212 between buses of different types. This clause describes how routing information is created, maintained,
1213 and used by MCTP bridges and MCTP endpoints. Keep the following key points in mind about MCTP
1214 bridges:

- 1215 • An MCTP bridge is responsible for routing MCTP packets between at least two buses.
- 1216 • An MCTP bridge is typically the bus owner for at least one of those buses.

1217 **9.1.1 Routing/bridging restrictions**

1218 Figure 7 and Figure 8 illustrate some of the supported and unsupported bridging topologies. As shown, it
1219 is acceptable for a given topology to have more than one path to get to a given EID. This can occur either
1220 because different media are used or because a redundant or failover communication path is desired in an
1221 implementation.

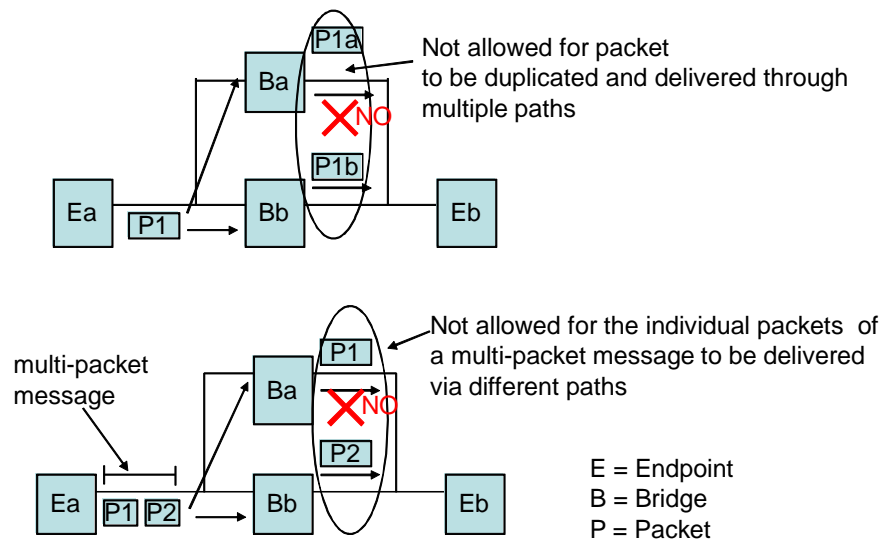
1222 A bridge shall not route or forward packets with a broadcast destination ID.



1223

1224

Figure 7 – Acceptable failover/redundant communication topologies



1225

1226

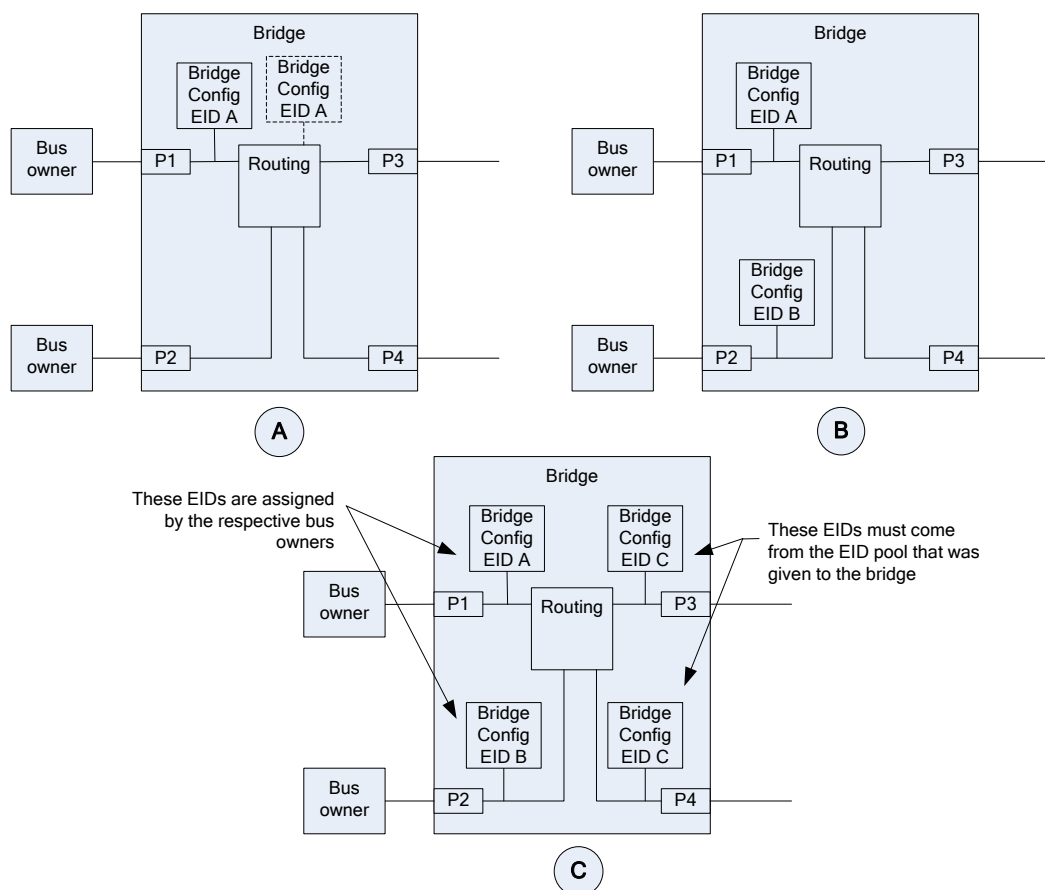
Figure 8 – Routing/bridging restrictions

1227 **9.1.2 EID options for MCTP bridges**

1228 An MCTP bridge that connects to multiple buses can have a single EID or multiple EIDs through which
 1229 the bridge's routing configuration and endpoint functionality can be accessed through MCTP control
 1230 commands. There are three general options:

- 1231 • The bridge uses a single MCTP endpoint
- 1232 • The bridge uses an MCTP endpoint for each bus that connects to a bus owner
- 1233 • The bridge uses an MCTP endpoint for every bus to which it connects

1234 Examples of these different options are shown in Figure 9, and more detailed information on the options
 1235 is provided following the figure.



1236

1237

Figure 9 – EID options for MCTP bridges

1238 A bridge has only one EID pool. To prevent issues with getting an EID pool allocation from multiple bus
 1239 owners, a bridge that is accessible through multiple EIDs will only accept EID pool allocation from the first
 1240 bus that allocation is received from using the Allocate Endpoint IDs command. This behavior is described
 1241 in more detail in the specification of the Allocate Endpoint IDs command.

1242 If necessary, the Get Endpoint UUID command can be used to correlate that EIDs belong to the same
 1243 MCTP bridge device. (This correlation is not required for normal initialization and operation of the MCTP
 1244 network, but it may be useful when debugging.)

1245 The following is a more detailed description of the different EID options for bridges:

1246 • **Single endpoint**

1247 A single endpoint is used to access the bridge's routing configuration and endpoint functionality.
1248 Referring to diagram (A) in Figure 9, an implementation may elect to either have the endpoint
1249 functionality be directly associated with a particular bus/port (for example, P1) or the
1250 functionality can be located on a "virtual bus" that is behind the routing function. In either case,
1251 the routing functionality ensures that the EID can be accessed through any of the buses to
1252 which the bridge connects.

1253 Although there is a single endpoint, the bridge shall report the need for EID assignment for that
1254 endpoint on each bus that is connected to a bus owner (for example, P1, P2). The multiple
1255 announcements provide a level of failover capability in the EID assignment process in case a
1256 particular bus owner becomes unavailable. The multiple announcements also help support a
1257 consistent EID assignment process across bus owners. To prevent issues with getting
1258 conflicting EID assignments from multiple bus owners, the bridge will only accept EID pool
1259 allocation from the first bus that an allocation is received from using the Set Endpoint ID
1260 command. This behavior is described in more detail in the specification of the Set Endpoint ID
1261 command. The bridge shall not report the need for EID assignment on any buses that the bridge
1262 itself owns.

1263 • **Endpoint for each bus connection to a bus owner**

1264 The bridge has one endpoint for each bus connected to a bus owner. This is shown as diagram
1265 (B) in Figure 9. There are no explicit endpoints associated with buses that are not connected to
1266 a bus owner (for example, the buses connected to ports P3 and P4, respectively.) Because of
1267 the way packet routing works, EID A and EID B can be accessed from any of the ports
1268 connected to the bridge. Thus, the bridge's configuration functionality may be accessed through
1269 multiple EIDs. Because a separate endpoint communication terminus is associated with each
1270 port (P1, P2), the bridge can accept an EID assignment for each bus independently.

1271 The bridge shall only report the need for EID assignment on buses that connect to a bus owner,
1272 and only for the particular MCTP control interface that is associated with the particular bus. For
1273 example, the bridge would announce the need for EID assignment for the interface associated
1274 with EID A only through P1, and the need for EID assignment for the interface associated with
1275 EID B only through P2. The bridge shall not report the need for EID assignment on any buses
1276 that the bridge itself owns.

1277 • **Endpoint for every bus connection**

1278 The bridge has one endpoint for each bus connected to it, as shown as diagram (C) in Figure 6.
1279 This includes buses that connect to bus owners (for example, P1, P2) and buses for which the
1280 bridge is the bus owner (for example, P3, P4). Because of the way packet routing works, any of
1281 these EIDs can be accessed from any of the ports connected to the bridge.

1282 Because a separate endpoint communication terminus is associated with each owned port (P1,
1283 P2), the bridge can accept an EID assignment for the bus owners of each bus independently.
1284 The EIDs associated with the buses that the bridge itself owns (for example, P3, P4) shall be
1285 taken out of the EID pool that is allocated to the bridge.

1286 The bridge shall only report the need for EID assignment on buses that connect to a bus owner,
1287 and only for the particular MCTP control interface that is associated with the particular bus. For
1288 example, the bridge would announce the need for EID assignment for the interface associated
1289 with EID A only through P1, and the need for EID assignment for the interface associated with
1290 EID B only through P2. The bridge shall not report the need for EID assignment on any buses
1291 that the bridge itself owns.

1292 9.1.3 Routing table

1293 An MCTP bridge maintains a routing table where each entry in the table associates either a single EID or
1294 a range of EIDs with a single physical address and bus ID for devices that are on buses that are directly
1295 connected to the bridge.

1296 If the device is a bridge, there will typically be a range of EIDs that are associated with the physical
1297 address of the bridge. There may also be an entry with a single EID for the bridge itself.

1298 9.1.4 Bridging process overview

1299 When a bridge receives an MCTP packet, the following process occurs:

- 1300 1) The bridge checks to see whether the destination EID in the packet matches or falls within the
1301 range of EIDs in the table.
- 1302 2) If the EID is for the bridge itself, the bridge internally consumes the packet.
- 1303 3) If there is a match with an entry in the routing table, the following steps happen:
 - 1304 • The bridge changes the physical addresses in the packet and reformats the medium-
1305 specific header and trailer fields as needed for the destination bus.
 - 1306 • The destination physical address from the source bus is replaced with the destination
1307 physical address for the destination bus obtained from the entry in the routing table.
 - 1308 • The bridge replaces the source physical address in the packet it received with the bridge's
1309 own physical address on the target bus. This is necessary to enable messages to be
1310 routed back to the originator.
 - 1311 • Packet-specific transport header and data integrity fields are updated as required by the
1312 particular transport binding.
- 1313 4) If there is no match, packets with EID values that are not in the routing table are silently
1314 discarded.

1315 9.1.5 Endpoint operation with bridging

1316 A bridge does not track the packet transmissions between endpoints. It simply takes packets that it
1317 receives and routes them on a per-packet basis based on the destination EID in the packet. It does not
1318 pay attention to message assembly or disassembly or message type-specific semantics, such as
1319 request/response semantics, for packets that it routes to other endpoints.

1320 Most simple MCTP endpoints will never need to know about bridges. Typically, another endpoint will
1321 initiate communication with them. The endpoint can then simply take the physical address and source
1322 EID information from the message and use that to send messages back to the message originator.

1323 An endpoint that needs to originate a "connection" to another MCTP endpoint does need to know what
1324 physical address should be used for messages to be delivered to that endpoint. To get this information, it
1325 needs to query the bus owner for it. An endpoint knows the physical address of the bus owner because it
1326 saved that information when it got its EID assignment.

1327 The Resolve Endpoint ID command requests a bus owner to return the physical address that is to be
1328 used to route packets to a given EID. (This is essentially the MCTP equivalent of the ARP protocol that is
1329 used to translate IP addresses to physical addresses.) The address that is returned in the Resolve
1330 Endpoint ID command response will either be the actual physical address for the device implementing the
1331 endpoint, or it will be the physical address for the bridge to be used to route packets to the desired
1332 endpoint.

1333 Because the physical address format is media-specific, the format of the physical address parameter is
 1334 documented in the specifications for the particular media-specific physical transport binding for MCTP (for
 1335 example, MCTP over SMBus/I2C, MCTP over PCIe Vendor Defined Messaging, and so on).

1336 If endpoint A has received a message from another endpoint B, it does not need to issue a Resolve
 1337 Endpoint ID command. Instead, it can extract the source EID and source physical address from the
 1338 earlier message from endpoint B, and then use that as the destination EID and destination physical
 1339 address for the message to Endpoint B.

1340 9.1.6 Routing table entries

1341 Each MCTP device that does bridging shall maintain a logical routing table. A bus owner shall also
 1342 typically maintain a routing table if more than one MCTP device is connected to the bus that it owns. The
 1343 routing table is required because the bus owner is also the party responsible for resolving EIDs to
 1344 physical addresses.

1345 The internal format that a device uses for organizing the routing table is implementation dependent. From
 1346 a logical point of view, each entry in a routing table will be comprised of at least three elements: An EID
 1347 range, a bus identifier, and a bus address. This is illustrated in Figure 10.

EID Range	Bus ID	Bus Address
-----------	--------	-------------

1348 **Figure 10 – Basic routing table entry fields**

1349 The *EID range* specifies the set of EIDs that can be reached through a particular bus address on a given
 1350 bus. Because the bus ID and bus address may correspond to a particular "port" on a bridge, it is possible
 1351 that there can be multiple non-contiguous ranges (multiple routing table entries) that have the same bus
 1352 ID/bus address pair route. EIDs and EID ranges can be categorized into three types: downstream,
 1353 upstream, and local. "Downstream" refers to EIDs that are associated with routing table entries that are
 1354 for buses that are owned by the bridge that is maintaining the routing table. "Upstream" refers to EIDs that
 1355 are associated with routing table entries that route to buses that are not owned by the bridge that is
 1356 maintaining the routing table.

1357 "Local" refers to the EIDs for routing table entries for endpoints that are on buses that are directly
 1358 connected to the bridge that is maintaining the routing table. A particular characteristic of entries for local
 1359 EIDs is that the Resolve Endpoint ID command is issued from the same bus that the endpoint is on. The
 1360 bridge/bus owner delivers the physical address for that endpoint rather than the physical address
 1361 associated with a routing function. This facilitates allowing endpoints on the same the bus to
 1362 communicate without having to go through an MCTP routing function.

1363 A routing table entry may not be "local" even if two endpoints are located on the same bus. An implementation may
 1364 require that different endpoints go through the routing function to intercommunicate even if the endpoints are part of
 1365 the same bus.

1366 The *bus ID* is an internal identifier that allows the MCTP device to identify the bus that correlates to this
 1367 route. MCTP does not require particular values to be used for identifying a given physical bus connection
 1368 on a device. However, this value will typically be a 0-based numeric value.

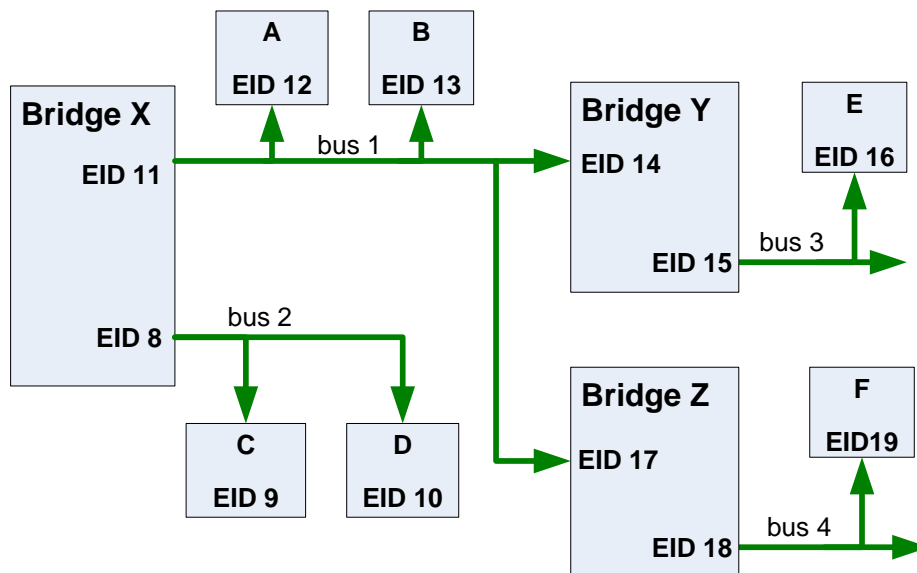
1369 EXAMPLE: A device that had three buses would typically identify them as buses "0", "1", and "2".

1370 The *bus address* is the physical address of a specific device on the bus through which the EIDs specified
 1371 in the *EID range* can be reached. This can either be the physical address corresponding to the
 1372 destination endpoint, or it can be the physical address of the next bridge in the path to the device. The
 1373 format of this address is specific to the particular physical medium and is defined by the physical medium
 1374 transport binding.

1375 **9.1.7 Routing table creation**

1376 This clause illustrates the types of routing information that a bridge requires, and where the information
 1377 comes from. This clause also describes the steps that a bus owner shall use to convey that information
 1378 for a given bus.

1379 Figure 11 helps illustrate the steps that are required to completely establish the routing information
 1380 required by a bridge (bridge Y). The arrows in Figure 11 point outward from the bus owner and inward to
 1381 "owned" endpoints on the bus.



1382

1383

Figure 11 – Routing table population

1384 **9.1.7.1 Routing table population example**

1385 With reference to Figure 11, the following items describe the information that bridge Y will need for routing
 1386 messages in the example topology shown:

- 1387
- 1388 • It needs a set of EIDs allocated to it to use for itself and to allocate to other devices (for
 1389 example, EIDs 14:16). These are allocated to it by the bus owner (bridge X).
 - 1390 • It needs a routing table that has an entry that maps EID 16 to the physical address for device E
 1391 on bus 3.
 - 1392 • It needs routing table entries for the local devices on bus 1, which are: bridge X (EID 11), device
 1393 A (EID 12), device B (EID 13), and bridge Z (EID 17), assuming that devices A and B are to be
 1394 reached by bridge Y without having to go through bridge X. This information shall be given to it
 by the bus owner (bridge X).

- 1395 • It needs to know that EIDs 8:10 are accessed through bus owner/bridge X. Therefore, it needs a
1396 routing table entry that maps the EID range 9:10 to the physical address for bridge X on bus 1.
1397 This information shall also be given to it by the bus owner (bridge X).
- 1398 • It needs to know that EIDs 17:19 are accessed through bridge Z. Therefore, it needs a routing
1399 table entry that maps the EID range 17:19 to the physical address for bridge Z on bus 1.
1400 Because the bus owner (bridge X) allocated that range of EIDs to bridge Z in the first place, this
1401 information is also given to bridge Y by the bus owner (bridge X).

1402 9.1.7.2 Bus initialization example

1403 Starting with the description of what bridge Y requires, the following task list shows the steps that bridge
1404 X shall take to provide routing information for bus 1. Bridge X shall:

- 1405 1) Assign EIDs to devices A, B, C, D, bridge Y, and bridge Z. This is done using the Set Endpoint
1406 ID command. The response of the Set Endpoint ID command also indicates whether a device
1407 wants an additional pool of EIDs.
- 1408 2) Allocate EID pools to bridge Y and bridge Z. This is done using the Allocate Endpoint IDs
1409 command.
- 1410 3) Tell bridge Y the physical addresses and EIDs for devices A and B, bridge X (itself), and bridge
1411 Z on bus 1. This is done using the Routing Information Update command.
- 1412 4) Tell bridge Y that EIDs 18:19 are accessed through the physical address for bridge Z on bus 1.
1413 This is also done using the Routing Information Update command. (Steps 3 and 4 can be
1414 combined and covered with one instance of the command.)
- 1415 5) Tell bridge Z the physical addresses and EIDs for devices A and B, bridge X (itself), and bridge
1416 Y on bus 1. This is also done using the Routing Information Update command.
- 1417 6) Tell bridge Z that EIDs 15:16 are accessed through the physical address for bridge Y on bus 1.
1418 This is also done using the Routing Information Update command. (Steps 5 and 6 can be
1419 combined and covered with one instance of the command.)
- 1420 7) Tell bridge Y and bridge Z that EIDs 8:10 are accessed through bridge X on bus 1. This is also
1421 done using the Routing Information Update command. This step could also be combined with
1422 steps 3 and 4 for bridge Y and steps 5 and 6 for bridge Z.

1423 9.1.8 Routing table updates responsibility for bus owners

1424 After it is initialized for all bridges, routing table information does not typically require updating during
1425 operation. However, updating may be required if a bridge is added as a hot-plug device. In this case,
1426 when the bridge is added to the system, it will trigger the need for the bus owner to assign it an EID,
1427 which will subsequently cause the request for EID pool allocations, and so on. At this time, the bus owner
1428 can simply elect to re-run the steps for bus initialization as described in 9.1.7.2.

1429 9.1.9 Consolidating routing table entries

1430 MCTP requires that when an EID pool is allocated to a device, the range of EIDs is contiguous and
1431 follows the EID for the bridge itself. Thus, a bridge can elect to consolidate routing table information into
1432 one entry when it recognizes that it has received an EID or EID range that is contiguous with an existing
1433 entry for the same physical address and bus. (The reason that EID allocation and routing information
1434 updates are not done as one range using the same command is because of the possibility that a device
1435 may have already received an allocation from a different bus owner.)

1436 **9.2 Bridge and routing table examples**

1437 The following examples illustrate different bridge and MCTP network configurations and the
 1438 corresponding information that shall be retained by the bridge for MCTP packet routing and to support
 1439 commands such as Resolve Endpoint ID and Query Hop.

1440 The following clauses (including Table 4 through Table 6) illustrate possible topologies and ways to
 1441 organize the information that the bridge retains. Implementations may elect to organize and store the
 1442 same information in different ways. The important aspect of the examples is to show what information is
 1443 kept for each EID, to show what actions cause an entry to be created, and to show how an EID or EID
 1444 range can in some cases map to more than one physical address.

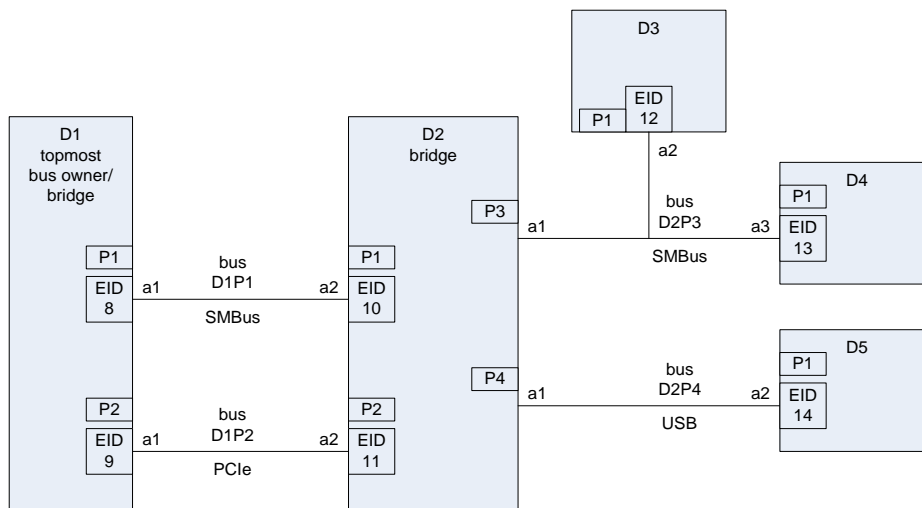
1445 The examples show a possible time order in which the entries of the table are created. Note that a given
 1446 implementation of the same example topology could have the entries populated in a different order. For
 1447 example, if there are two bus owners connected to a bridge, there is no fixed order that the bus owners
 1448 would be required to initialize a downstream bridge. Additionally, there is no requirement that bus owners
 1449 perform EID assignment or EID pool allocation in a particular order. One implementation may elect to
 1450 allocate EID pools to individual bridges right after it has assigned the bridge its EID. Another
 1451 implementation may elect to assign all the EIDs to devices first, and then allocate the EID pools to
 1452 bridges.

1453 **9.2.1 Example 1: Bridge D2 with an EID per "Owned" port**

1454 Figure 12 shows the routing table in a bridge (D2), where D2 has an EID associated with each bus
 1455 connected to a bus owner. In this example, D1 is not implementing any internal bridging between its P1
 1456 and P2. Consequently, EID2 cannot be reached by bridging through EID1 and vice versa (see Table 4).

1457 NOTE: If there was internal bridging, D1 would need to provide routing information that indicated that EID2 was
 1458 reachable by going through EID1 and vice versa. In this case, D1 would provide routing information that EID range
 1459 (EID1...EID2) would be accessed through D1P1a1 on SMBus and D1P1a2 on PCIe.

1460 **Key: D = device, P = port, a = physical address**




1461

1462 **Figure 12 – Example 1 Routing topology**

1463

Table 4 – Example 1 Routing table for D2

Time	EID	EID Access Port	Medium Type	Access Physical Address	Device/Entry Type	Entry Was Created and Populated By
	EID 10	P1	SMBus	D1P1a2	Bridge, Self	Self when EID was assigned by D1
	EID 11	P2	PCIe	D1P2a2	Bridge, Self	Self when EID was assigned by D1
	EID 12	P3	SMBus	D2P3a2	Endpoint	Self after D1 assigned EID pool (typically the entry will not be created until after the bridge D2 assigns EID 12 to D3)
	EID 13	P3	SMBus	D2P3a3	Endpoint	Self after D1 assigned EID pool (typically the entry will not be created until after the bridge D2 assigns EID 13 to D4)
	EID 14	P4	USB	D2P4a2	Endpoint	Self after D1 assigned EID pool (typically the entry will not be created until after the bridge D2 assigns EID 14 to D5)
	EID 8	P1	SMBus	D1P1a1	Bridge	D1 through Routing Information Update command
	EID 9	P2	PCIe	D1P2a1	Bridge	D1 through Routing Information Update command

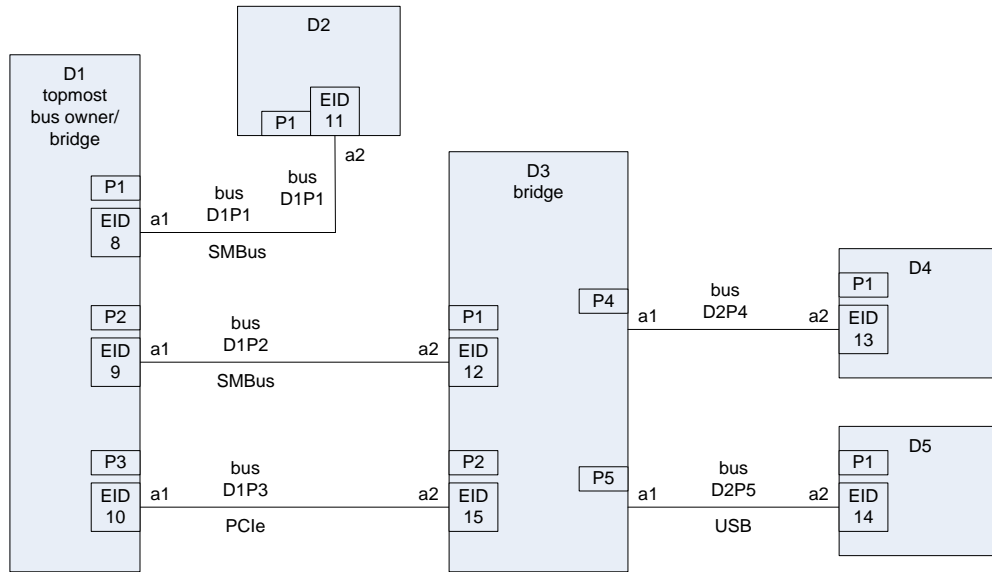
1464 **9.2.2 Example 2: Topmost bus owner D1**

1465 Figure 13 assumes the following conditions:

- 1466 • D1 assigns its internal EIDs first.
- 1467 • The buses are handled in the order D1P1, D1P2, D1P3.
- 1468 • D1 allocates the EID pool to bridges right after it has assigned the EID to the device.

1469 Similar to Example 1, this example assumes that there is no internal bridging within D1 between P1, P2,
 1470 and P3. This scenario is reflected in Table 5.

1471 Key: D = device, P = port, a = physical address



1472

1473

Figure 13 – Example 2 Routing topology

1474

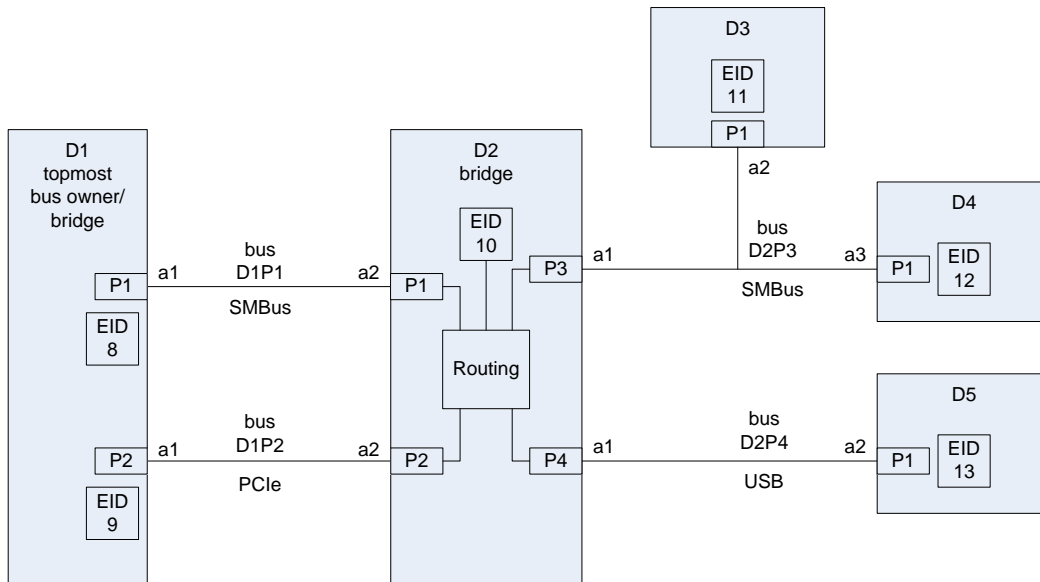
Table 5 – Example 2 Routing table for D1

EID	EID Access Port	Medium Type	Access Physical Address	Device/Entry Type	Entry Was Created and Populated By
EID 8	P1	SMBus	D1P1a1	Bridge, self	Self
EID 9	P2	SMBus	D1P2a1	Bridge, self	Self
EID 10	P3	PCIe	D1P3a1	Bridge, self	Self
EID 11	P1	SMBus	D1P1a2	Endpoint	Self upon assigning EID to device D2
EID 12	P2	SMBus	D1P2a2	Bridge	Self upon assigning EID 5 to bridge D3
EID 13:14	P2	SMBus	D1P2a2	Bridge pool	Self upon assigning EID pool to bridge D3
EID 15	P3	PCIe	D1P3a2	Bridge	Self upon assigning EID 8 to bridge D3
EID 13:14	P3	PCIe	D1P3a2	Bridge pool	Self upon issuing an Allocate Endpoint IDs command and finding that bridge D3 already has an assigned pool, D1 creates this entry by extracting the EIDs for this entry from the response to the Allocate Endpoint IDs command

1475 **9.2.3 Example 3: Bridge D2 with single EID**

1476 Figure 14 assumes that bridge D2 has a single EID and gets its EID assignment and EID allocation
 1477 through bus D1P1 first, and that bus D1P2 later gets initialized. This scenario is reflected in Table 6.

1478 Key: D = device, P = port, a = physical address



1479

1480

Figure 14 – Example 3 Routing topology

1481

Table 6 – Example 3 Routing table for D2

Target EID	Target Endpoint Access Port	Target EID Access Physical Address	Device/Entry Type	Entry Was Created and Populated By
EID 10	P1	D1P1a2	Bridge, self	All four entries created by self (bridge) upon receiving initial EID assignment from D1 through P1
EID 10	P2	D1P2a2	Bridge, self	
EID 10	P3	D2P3a1	Bridge, self	
EID 10	P4	D2P4a1	Bridge, self	
EID 11	P3	D2P3a2	Endpoint	Self after D1 allocated EID pool (typically the entry will not be created until after the bridge D2 assigns EID 11 to D3)
EID 12	P3	D2P3a3	Endpoint	Self after D1 allocated EID pool (typically the entry will not be created until after the bridge D2 assigns EID 12 to D4)
EID 13	P3	D2P4a2	Endpoint	Self after D1 allocated EID pool (typically the entry will not be created until after the bridge D2 assigns EID 13 to D5)
EID 8:9	P1	D1P1a1	Bridge	D1 through Routing Information Update command
EID 8:9	P2	D1P2a1	Bridge	D1 through Routing Information Update command

1482 **9.2.4 Additional information tracked by bridges**

1483 In addition to the information required to route messages between different ports, a bridge has to track
 1484 information to handle MCTP control commands related to the configuration and operation of bridging
 1485 (shown in Table 7).

1486 **Table 7 – Additional information tracked by bridges**

What	Why
Which buses are connected to a bus owner	This information tells the bridge from which buses it should request EID assignment. This will typically be accomplished as a non-volatile configuration or hardware-strapping option for the bridge.
Which bus the bridge received its EID assignment through the Set Endpoint ID command	If the bridge uses a single EID that is shared across multiple "owned" buses, this information is used to track which bus the request came in on, so that the bridge can reject EID assignment requests from other buses.
Which bus it received the Routing Information Update command from for creating a particular routing table entry	This information is required so that if a future Routing Information Update command is received, the bridge will update only the entries corresponding to that bus.
Which bus it received its EID pool allocation from through the Allocate Endpoint IDs command	This information is used to track which bus the request came in on so that the bridge can reject EID pool allocations from other buses.
The physical medium and physical addressing format used for each port	This information is used to provide the correctly formatted response to commands such as Resolve Endpoint ID and for bridging MCTP packets between the different buses that the bridge supports. Because this is related to the physical ports and hardware of the bridge, this information will typically be "hard coded" into the bridge.

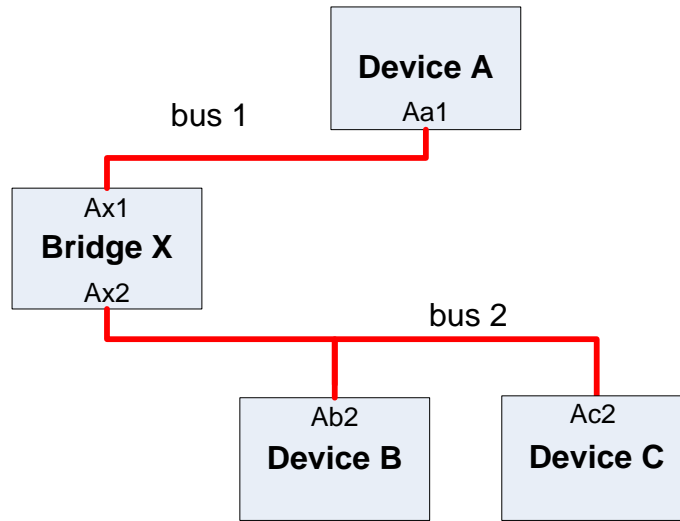
1487 **9.3 Endpoint ID resolution**

1488 When a device uses the Resolve Endpoint ID command to request the resolution of a given endpoint to a
 1489 physical address, the bridge shall respond based on which bus the request came in on.

1490 For example, consider Figure 15. If device A wishes to get the physical address needed to send a
 1491 message to device C, it sends a Resolve Endpoint ID command to bus owner bridge X through address
 1492 Ax1. Because device A shall go through bridge X to get to device C, bridge X responds with its physical
 1493 address Ax1.

1494 When device B wishes to know the address to use to communicate with device C, it sends a Resolve
 1495 Endpoint ID request to bridge X through address Ax2. In this case, bridge X can respond by giving device
 1496 B the direct physical address of device C on bus 2, Ac2.

1497 Thus, the Resolve Endpoint ID command can return a different response based on the bus from which
 1498 the Resolve Endpoint ID command was received.



notation:
 Ab2 = physical Address of device b on bus 2.

1499

1500

Figure 15 – Endpoint ID resolution

1501 **9.3.1 Resolving multiple paths**

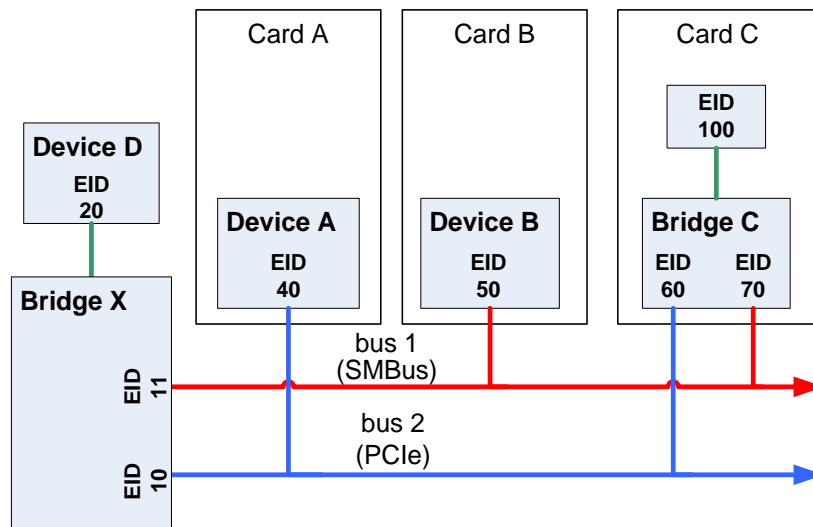
1502 Cases can occur where there can be more than one possible path to a given EID. A likely scenario is
 1503 shown in Figure 16. In Figure 16, assume that the system topology supports cards that connect to either
 1504 SMBus, PCIe, or both. Bridge X is the bus owner for both buses.

1505 NOTE: This is a logical representation of MCTP buses. Physically, the buses may be formed of multiple physical
 1506 segments, as would be the case if one of the MCTP buses was built using PCIe.

1507 As shown, card C contains a bridge that connects to both buses. Thus, the device with EID 100 can be
 1508 reached either from bus 1 or bus 2.

1509 If device D wishes to send a message to EID 100, bridge X can choose to route that message either
 1510 through bus 1 or bus 2. MCTP does not have a requirement on how this is accomplished. The general
 1511 recommendation is that the bridge preferentially selects the faster available medium. In this example, that
 1512 would be PCIe.

1513 NOTE There are possible topologies where that simple rule may not yield the preferred path to a device. However,
 1514 in most common implementations in PC systems, this approach should be effective. A vendor making a bridge device
 1515 may consider providing configuration options to enable alternative policies.



1516

1517

Figure 16 – Resolving multiple paths

1518 9.4 Bridge and bus owner implementation recommendations

1519 This clause provides recommendations on EID pool and routing table sizes for devices that implement
1520 bridge and bus owner functionality.

1521 9.4.1 Endpoint ID pool recommendations

1522 The system design should seek to minimize the number of devices that need to allocate EID pools to hot-
1523 plug devices or add-in cards. If feasible, the system design should have all busses that support hot-plug
1524 devices/add-in cards owned by a single device.

1525 If only one device handles the hot-plug devices and add-in cards, it will be simpler for the system
1526 integrator to configure devices and allocate EID pools. Because any other bridges in the system that do
1527 not handle hot-plug devices only need to handle a fixed number of MCTP devices, it will be known at
1528 design time how large an EID pool will be required. The remaining number of EIDs can then simply be
1529 allocated to the single device that handles the hot-plug devices and add-in cards.

1530 To support this, it is recommended that devices that operate as bridges include a non-volatile
1531 configuration option that enables the system integrator to configure the size of the EID pool they request.

1532 9.4.2 Routing table size recommendations

1533 This clause provides some initial recommendations and approaches on how to determine what target
1534 routing table entry support to provide in a device.

- 1535 • **PCIe slots**

1536 To provide entries to support devices that plug into PCIe slots, assume that each slot may
1537 support both PCIe and SMBus endpoints and provide support for at least two endpoints per bus
1538 type.

1539 This means providing support for at least four directly connected endpoints per card. (Other
1540 endpoints may be behind bridges on the card, but this does not affect the routing table size for
1541 the bus owner.) This implies at least four routing table entries per PCIe slot. Thus, a device that
1542 was designed to support system implementations with eight PCIe slots should have support for
1543 32 routing table entries.

1544 • **Planar PCIe devices**
 1545 In most PC systems, PCIe would be typically implemented as a single MCTP bus owned by a
 1546 single device as the bus owner. Thus, the number of static devices should be proportional to the
 1547 number of PCIe devices that are built into the motherboard.

1548 Typically, this is fewer than eight devices. Thus it is recommended to support at least eight
 1549 entries for static PCIe devices.

1550 • **Static SMBus/I2C MCTP devices**

1551 The routing table should also be sized to support an additional number of "static" devices on
 1552 owned buses. At this time, it is considered unlikely that more than a few MCTP devices would
 1553 be used on a given SMBus/I2C bus. Most devices would be non-intelligent sensor and I/O
 1554 devices instead. Conservatively, it is recommended that at least four entries be provided for
 1555 each SMBus/I2C bus that the device owns.

1556 Example 1: "client" capable device
 1557 Four PCIe slots → 16 routing table entries
 1558 Two owned SMBus/I2C busses → +8 entries
 1559 Static PCIe device support → +8 entries
 1560 ~32 entries or more

1561 Example 2: volume server capable
 1562 Eight PCIe slots → 32 routing table entries
 1563 Four owned SMBus/I2C busses → +16 entries
 1564 Static PCIe device support → +8 entries
 1565 ~56 entries or more

1566 **9.5 Path and transmission unit discovery**

1567 The transmission unit is defined as the size of the MCTP packet payload that is supported for use in
 1568 MCTP message assembly for a given message. The supported transmission unit sizes are allowed to
 1569 vary on a per-message type basis.

1570 Intermediate bridges and physical media can limit the transmission unit sizes between endpoints.
 1571 Therefore, the MCTP control protocol specifies a mechanism for discovering the transmission unit support
 1572 for the path between endpoints when one or more bridges exist in the path between the endpoints.

1573 The mechanism for path transmission unit discovery also enables the discovery of the bridges and
 1574 number of "hops" that are used to route an MCTP packet from one endpoint to another.

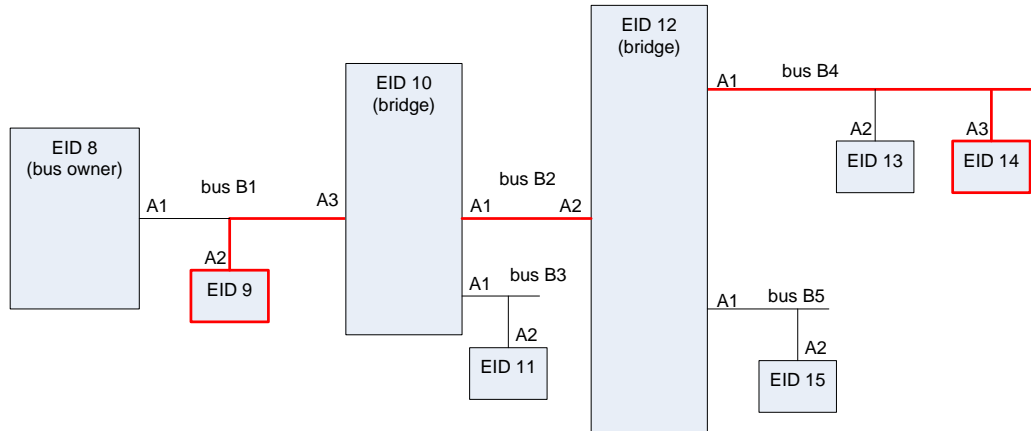
1575 **9.5.1 Path transmission unit negotiation**

1576 The MCTP control protocol only specifies how to discover what the path transmission unit size is for the
 1577 path between endpoints. The MCTP control protocol does not specify a generic mechanism for
 1578 discovering what transmission unit sizes a particular endpoint supports for a given message type.
 1579 Discovery and negotiation of transmission unit sizes for endpoints, if supported, is specified by the
 1580 definition of the particular message type.

1581 **9.5.2 Path transmission unit discovery process overview**

1582 This clause describes the process used for path transmission unit discovery. The discovery process
 1583 described here is designed to enable one endpoint to discover the path and transmission unit support for
 1584 accessing a particular "target" endpoint. It does not define a general mechanism for enabling an endpoint
 1585 to discover the path between any two arbitrary endpoints. For example, referring to
 1586 Figure 17, the process defines a way for the endpoint at EID 9 to discover the path/transmission unit

1587 support on the route to endpoint at EID 14, but this process does not define a process for EID 9 to
 1588 discover the path/transmission unit support between EID 11 and EID 14.



1589

1590

Figure 17 – Example path routing topology

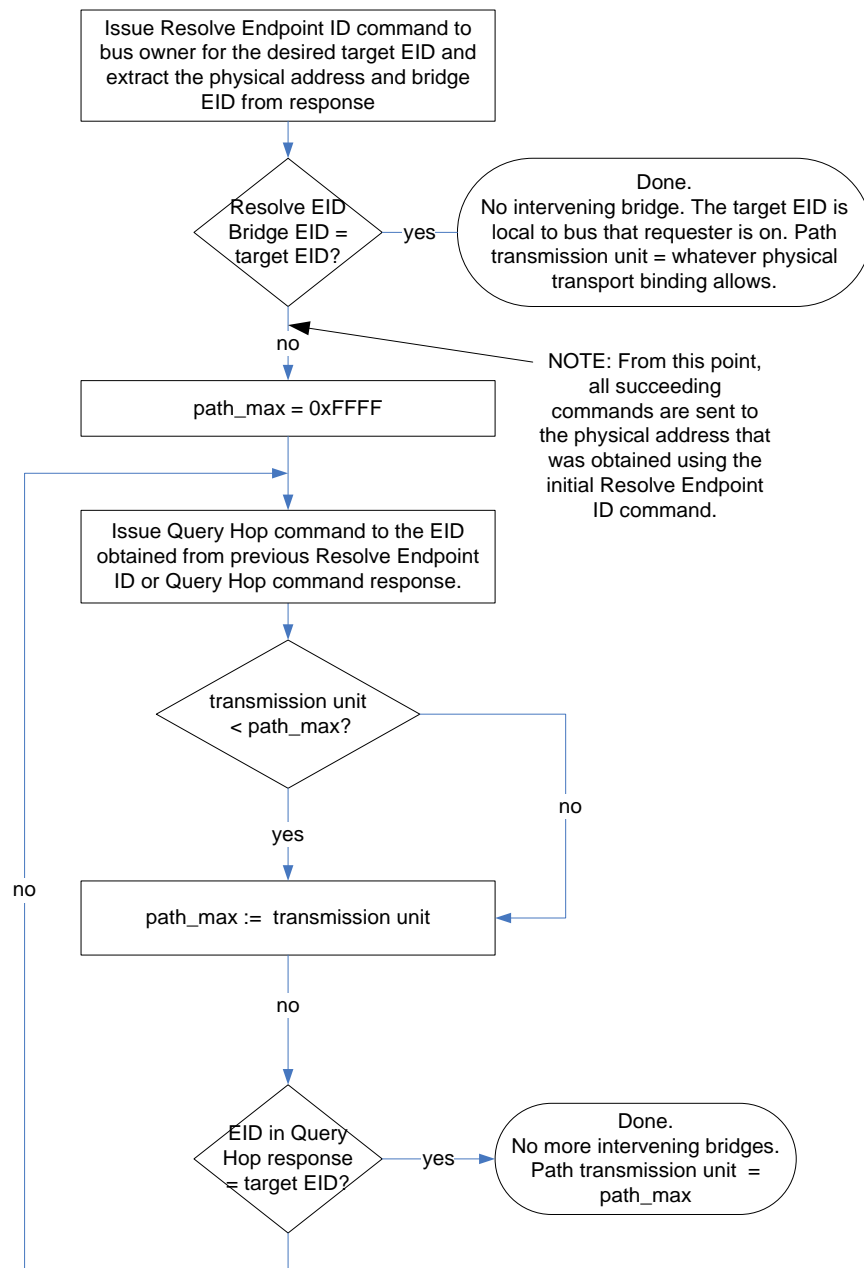
1591 The following example provides an overview of the path/transmission unit discovery process. The
 1592 example presumes that the MCTP network has already been initialized. Referring to
 1593 Figure 17, the endpoint with EID 9 wishes to discover the path used to access the endpoint with EID 14.
 1594 This discovery is accomplished using just two commands, Resolve Endpoint ID and Query Hop, as
 1595 follows:

- 1596 1) EID 9 first issues a Resolve Endpoint ID command to the bus owner, EID 8, with EID 14 as the
 1597 EID to resolve.
- 1598 2) EID 8 returns the physical address and EID of the bridge, EID 10 in the Resolve Endpoint ID
 1599 command response.
- 1600 3) EID 9 queries the bridge, EID 10, using a Query Hop command with EID 14 (the "target" EID) as
 1601 the request parameter. Note that EID 2 does not need to do another Resolve Endpoint ID
 1602 command because it already received the physical address of EID 3 from the original Resolve
 1603 Endpoint ID command.
- 1604 4) Bridge EID 10 responds to the Query Hop command by returning EID 12, which is the EID of the
 1605 next bridge required to access EID 14. The bridge EID 10 also returns the transmission unit
 1606 support that it offers for routing to the target EID.
- 1607 5) EID 9 then sends a Query Hop command to the bridge at EID 12. Note that EID 9 does not need
 1608 to do another Resolve Endpoint ID command because it already received the physical address
 1609 of EID 12 from the original Resolve Endpoint ID command.
- 1610 6) Bridge EID 12 responds to the Query Hop command by returning EID 14, which, because it is
 1611 the EID of the target endpoint, tells EID 9 that bridge EID 12 was the last "hop" in the path to
 1612 EID 6. The bridge EID 5 also returns the transmission unit support that it offers for routing to the
 1613 target EID.
- 1614 7) At this point, the bridges in the path to EID 14 have subsequently been discovered and their
 1615 respective transmission unit support returned. The effective transmission unit support for the
 1616 path to EID 14 will be the lesser of the transmission unit support values returned by the two
 1617 bridges.

1618 9.5.3 Path transmission unit discovery process flowchart

1619 The following flowchart (Figure 15) shows a generic algorithm for discovering the bridges in the path from
1620 one endpoint to a given target endpoint and the path transmission unit support. The flowchart has been
1621 intentionally simplified. Note that while the Query Hop command actually supports returning separate
1622 transmission unit sizes for the transmit and receive paths, the flowchart is simplified for illustration
1623 purposes and just refers to a single transmission unit for both transmit and receive.

1624 Additionally, Figure 18 does not show any explicit steps for error handling nor the process of handling
1625 command retries. In general, errors are most likely due to either an invalid EID being sent to the bridge
1626 (perhaps due to a programming error at the requester) or the EID not being present in the bridge's routing
1627 table. The latter condition could occur under normal operation if the requester did not realize that a
1628 routing table update had occurred because of a hot-plug update, for example. This error condition would
1629 be indicated by the bridge responding with an `ERROR_INVALID_DATA` completion code.



1630

1631

Figure 18 – Path transmission unit discovery flowchart

1632

9.6 Path transmission unit requirements for bridges

1633

An MCTP bridge routes packets between different buses, but it does not typically interpret the packet

1634

payload contents nor does it do assembly of those packets. Exceptions to this are when the bridge is

1635

handling packets addressed to its own EID, receives a Broadcast EID, and if the bridge supports different

1636

transmission units based on message type. See Table 32 for more information.

1637 10 Rate limiting

1638 Some MCTP bindings provide a significant transfer rate that may not be sustainable by the MCTP
1639 message receiver. It is not always possible to use the native flow control mechanisms of the medium,
1640 since they may be shared with other traffic. In order to help address this problem, Endpoints may support
1641 the following specified MCTP Rate Limiting method.

1642 Note; The PCIe binding is a typical example of this issue. PCIe provides significantly more bandwidth
1643 than most MCTP endpoints can consume. PCIe credits cannot be used to throttle the MCTP traffic, since
1644 this would throttle all PCIe traffic (MCTP and non-MCTP) to the device. Thus, an alternative Rate Limiting
1645 mechanism is needed. Rate limiting is performed independently in each direction and is not required to be
1646 symmetric. Rate limiting can be set for one-direction only, for both directions or not be set at all.

1647 The MCTP rate limit mechanism allows an endpoint on a specific medium to:

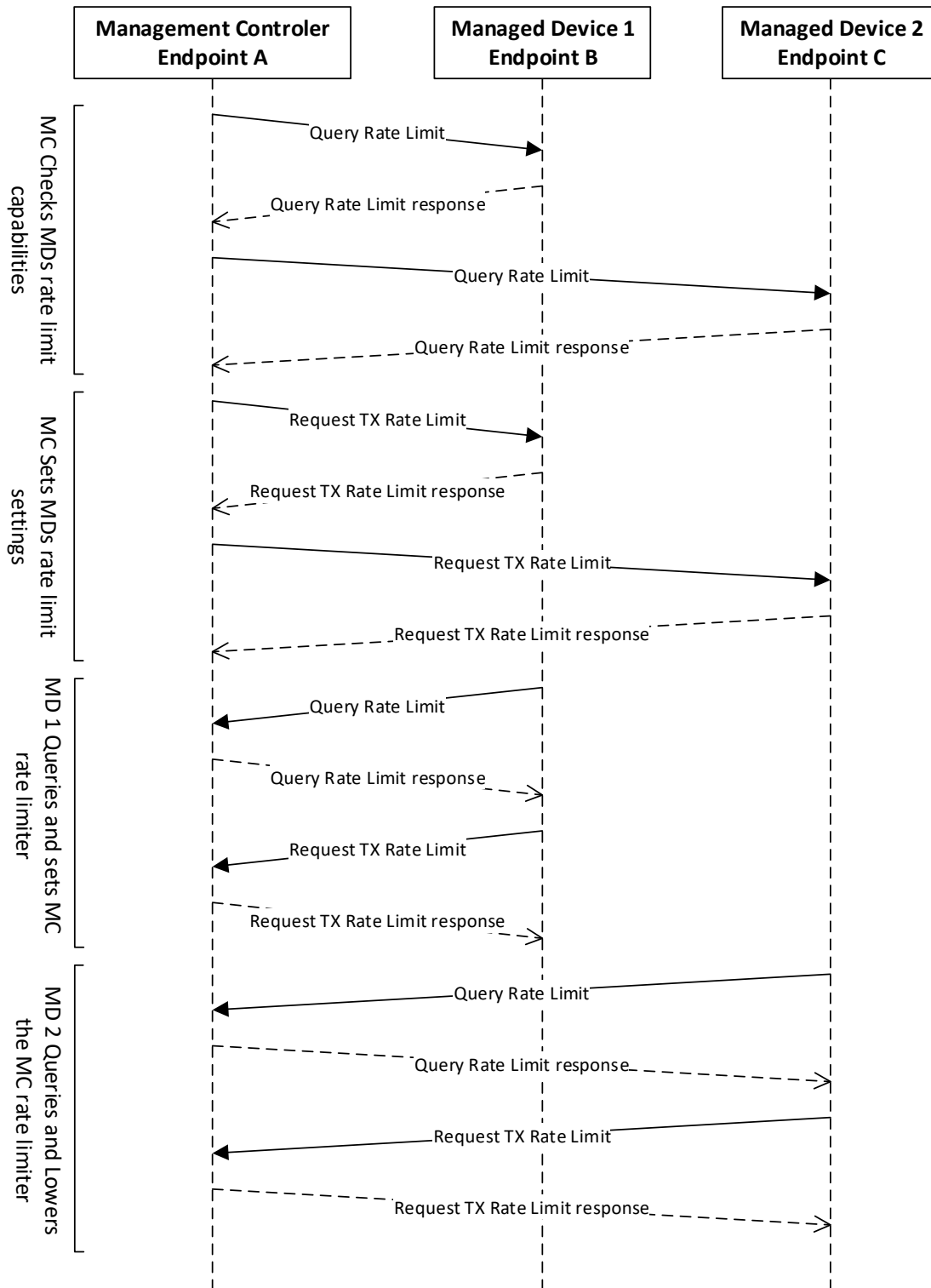
- 1648 • Publish its input processing rate and whether it can rate limit its output
- 1649 • Request its partner to rate limit its MCTP output traffic.

1650 Rate Limiting is negotiated between two endpoints and is configured on a per-EID basis such that devices
1651 having multiple EIDs should separately negotiate their Rate Limiting for each EID which supports Rate
1652 Limiting. If there are any MCTP Bridges in the path between the endpoints, the negotiated rate limit
1653 between the endpoints may not take bridge performance into account. The negotiation should take the
1654 speeds of the media for the path between the endpoints into account. Rate limiting is not specified for the
1655 bridging functionality within an MCTP Bridge (the functionality that routes MCTP packets between
1656 different ports on the bridge).

1657 Figure 19 presents an example of message exchanges for rate limiting. In this example, the management
1658 controller (MC) wants the managed devices (MD 1, MD 2) to send data at a limited rate. The MC first
1659 queries the MDs for their rate limiting capabilities using the Query Rate Limit command. Based on those
1660 capabilities, the MC requests the maximal transmit rate configuration for the MDs using the Request TX
1661 rate limit command. Conversely, the MDs may want to limit the rate that they receive data from the MC. In
1662 this case, it's the MDs that query the MC using the Query Rate Limit command, and, based on the
1663 response from the MC, requests configuration for the transmit rate from the MC using the Request TX
1664 rate limit command.

1665 Note that the figure does not show conditions such as handling the situation where one or more of the
1666 endpoints does not support Rate Limiting, nor does it show any algorithms that the endpoints may use to
1667 determine the best end-to-end value for Rate Limiting. Devices that negotiate Rate Limiting may wish to
1668 include algorithms or tests that would indicate there are intermediate devices in the path, such as
1669 Bridges, that would require transmit rates to be set to values that are lower than just what the receiving
1670 device needs. For example, the receiving device may detect that additional Rate Limiting is needed by
1671 noticing that there are packets missing in a multi-packet MCTP message transfer sequence.

1672



1673

1674

Figure 19 – Example rate limiting message exchanges

1675 10.1.1 Restrictions on rate limiting

1676 Message-based flow control may not utilize rate limiting. When rate limiting is active on a device which
1677 sends non-requested messages, then request/responses may also be affected by the rate limiting. Rate-
1678 limiting capable device may use rate limiting only to non-requested messages or to all messages. The
1679 transmit rate limiting operation-mode capability is reported by the device through "Transmit Rate Limiting
1680 operation capability" bit in Query rate limit command response.

1681 The use of rate limiting shall not supersede the timing requirements that are called out in other
1682 specifications, such as the transport binding specifications. Rate limiting shall include configuration
1683 options that allow meeting timing requirements under nominal operating conditions.

1684 10.1.2 Rate definition

1685 Let B be the Maximum supported burst size and R be the Maximum output rate limit in Packets Per
1686 Second (PPS), then the traffic shall be throttled such that in **any** time window $W = B/R$ (where $B \geq$
1687 1) there are no more than B packets.

1688 10.1.3 Output rate limiting capabilities parameters

1689 A transmitter that supports rate limiting shall expose its rate limiting capabilities using the Query Rate
1690 Limit command. For the definition of rate limiting, a baseline-transmission packet includes the baseline
1691 transmission unit as well as any medium-specific header/trailer and MCTP transport header. This
1692 includes:

- 1693 • **Maximum output rate limit:** The maximum rate in baseline transmission unit Packets/sec that
1694 the transmitting endpoint can be limited to when sending data to another endpoint.
- 1695 • **Minimum output rate limit:** The minimum rate in baseline transmission unit Packets/sec that
1696 the transmitting endpoint can be limited to when sending data to another endpoint. This value is
1697 also used to define the granularity of the configurable rate limit values.
- 1698 • **Maximum supported burst size:** The maximum number of consecutive baseline transmission
1699 unit Packets that the transmitter endpoint can send with minimal delay between MCTP packets.

1700 10.1.4 Input processing capabilities parameters

1701 A receiver can expose its input processing capabilities using the Query Rate Limit command. These
1702 parameters are informative only and should not be used to set the rate limiter of the partner. These
1703 parameters are intended to be used for visibility on the transmitter side, for performance analysis and
1704 monitoring purpose.

1705 The parameters exposed are:

- 1706 • **Maximum allowed receive data rate:** The maximum processing rate in baseline transmission
1707 unit packets/sec that the receiving endpoint can typically process incoming traffic. The data rate
1708 is measured using a time window. This rate is defined regardless of the content being received.
1709 Thus, devices which are limited in message processing shall report the maximum allowed
1710 receive data rate for minimal-size packets.
- 1711 • **Buffer Size:** this parameter defines the receive **buffer size** in bytes of the receiving endpoint.

1712 10.1.5 Rate limiting configuration parameters

1713 Rate limiting requirements are defined explicitly for each endpoint by means of two parameters, the
1714 maximum allowed data rate and the maximum continuous burst size. These are defined as follows:

- 1715 • **Maximum continuous burst size:** The maximum continuous burst size is defined in MCTP
1716 packets. Typically, this parameter reflects the receive buffer resources of the receiving endpoint.
- 1717 • **Maximum allowed data rate:** The maximum allowed data rate is defined in baseline
1718 transmission unit packets/sec. Typically, this defines the rate at which a receiving endpoint can
1719 process incoming messages. The data rate is measured using a time window as defined above.
1720 This rate is defined regardless of the content being received. Thus, devices which are limited in
1721 message processing shall request the maximum allowed transmit data rate with Burst Size of 1
1722 packet.

1723 If a device contains more than one MCTP endpoint (for example, a device that has an endpoint on
1724 SMBus/I2C and one on PCIe VDM) and supports setting rate limiting on these endpoints, then each rate-
1725 limiting configuration shall be independent and separately configurable. A device may include rate limiting
1726 capability for part or all of the endpoints.

1727 These parameters are used both by the receiver to request a specific traffic rate from the transmitter
1728 device and by the transmitter device to report the current rate-limiting values.

1729 When different settings are requested from different receiving endpoints, a transmitting endpoint that
1730 implements a single rate limiter shall use the smallest continuous burst size and the lowest data-rate that
1731 has been requested across the set of receiving endpoints. In a case of a single rate limiter, when traffic to
1732 multiple EIDs is active at the same time, the effective data rate to each of the receiving EIDs may be
1733 lower than the configured rate, as the aggregated data rates to all receiving EIDs will be the configured
1734 Rate Limiting settings.

1735 When a system is designed with devices supporting rate-limiting and devices which do not support rate-
1736 limiting, any device which supports rate limiting shall set its rate limiter to the negotiated rate-limiting
1737 settings. It is recommended that devices which do not support rate limiting are configured such that they
1738 will not cause buffer-overflow or data-processing rate overflow to their connected receiving endpoint. The
1739 implementation method of such a system is outside the scope of this specification.

1740 10.1.5.1 Updating rate-limiting parameters

1741 If an endpoint device needs to update the rate limiting settings of the other endpoint devices which are
1742 communicating with it and which are configured with rate limiting, it shall request the new settings in the
1743 sending devices using Request TX rate limit command. Once the response to Request TX rate limit
1744 command is received, the new rate limit is set according to the settings provided in the response. When
1745 the rate limiting settings is changed by an endpoint, the transmitting endpoint should notify the other
1746 receiving endpoint, sharing the same rate limiter, about the update using the Update rate limit command.

1747 11 MCTP control protocol

1748 MCTP control messages are used for the setup and initialization of MCTP communications within an
1749 MCTP network. This clause defines the protocol and formatting used for MCTP control messages over
1750 MCTP.

1751 11.1 Terminology

1752 The terms shown in Table 8 are used when describing the MCTP control protocol.

1753

Table 8 – MCTP control protocol terminology

Term	Description
Requester	The term “requester” is used to refer to the endpoint that originates an MCTP control Request message.
Responder	The term “responder” is used to refer to the endpoint that originates an MCTP control response message (that is, an endpoint that returns the response to an MCTP control Request message).
Originator or Source	The term “originator” or “source” is used to refer to the endpoint that originates any MCTP control message: Request, Response, or Datagram.
Target or Destination	The term “target” or “destination” is used to refer to the endpoint that is the intended recipient of any MCTP control message: Request, Response, or Datagram.
Asynchronous Notification	The term “asynchronous notification” is used to refer to the condition when an MCTP endpoint issues an un-requested Datagram to another MCTP endpoint.
Broadcast	The term “broadcast” is used when an MCTP control Datagram is sent out onto the bus using the broadcast EID.

1754 **11.1.1 Control message classes**

1755 The different types of messages shown in Table 9 are used under the MCTP control message type.

1756

Table 9 – MCTP control message types

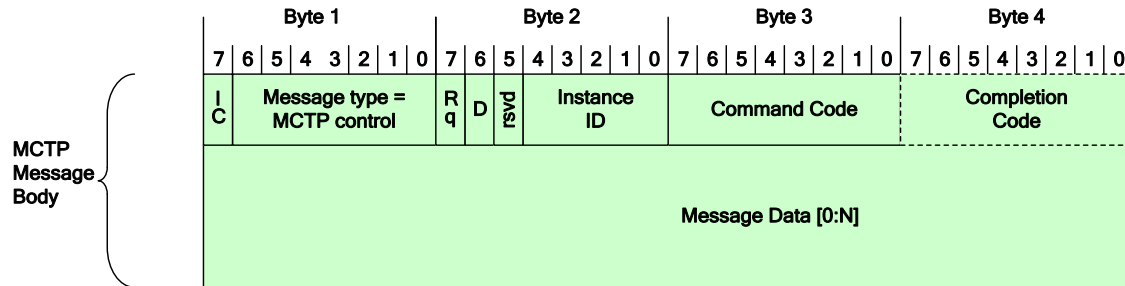
Type	Description
Request	This class of control message requests that an endpoint perform a specific MCTP control operation. All MCTP control Request messages are acknowledged with a corresponding Response message. (Within this specification, the term “command” and “request” are used interchangeably as shorthand to refer to MCTP control Request messages.)
Response	This class of MCTP control message is sent in response to an MCTP control Request message. The message includes a “Completion Code” field that indicates whether the response completed normally. The response can also return additional data dependent on the particular MCTP control Request that was issued.
Datagram	Datagrams are “unacknowledged” messages (that is, Datagrams do not have corresponding Response messages). This class of MCTP control message is used to transfer messages when an MCTP control Response message is neither required nor desirable.
Broadcast Request	A broadcast message is a special type of Request that is targeted to all endpoints on a given bus. All endpoints that receive the message are expected to interpret the Request.
Broadcast Datagram	A Datagram that is broadcast to all endpoints on the bus. Broadcast Datagrams are “unacknowledged” messages (that is, broadcast Datagrams do not have corresponding Response messages).

1757 **11.2 MCTP control message format**

1758 MCTP control messages use the MCTP control message type (see Table 3). Any message sent with this
 1759 message type will correspond to the definitions set forth in this clause. The basic format of an MCTP
 1760 control message is shown in Figure 20. Note that the byte offsets shown in Figure 20 are relative to the
 1761 start of the MCTP message body rather than the start of the physical packet.

1762 **11.2.1 Use of Message Integrity Check**

1763 MCTP control messages do not use a Message Integrity Check field. Therefore, the IC bit in MCTP
 1764 control messages shall always be 0b.



1765

1766 **Figure 20 – MCTP control message format**

1767 **11.3 MCTP control message fields**

1768 Table 10 lists the common fields for MCTP control messages.

1769

Table 10 – MCTP control message fields

Field Name	Description
IC*	Message Integrity Check bit = 0b. MCTP control messages do not include an overall Message Integrity check field.
Message Type*	MCTP control = 0x00 (000_0000b). This field identifies the MCTP message as being an MCTP control message.
Rq bit	Request bit. This bit is used to help differentiate between MCTP control Request messages and other message classes. Refer to 11.5.
D-bit	Datagram bit. This bit is used to indicate whether the Instance ID field is being used for tracking and matching requests and responses, or is just being used to identify a retransmitted message. Refer to 11.5.
Instance ID	The Instance ID field is used to identify new instances of an MCTP control Request or Datagram to differentiate new requests or datagrams that are sent to a given message terminus from retried messages that are sent to the same message terminus. The Instance ID field is also used to match up a particular instance of an MCTP Response message with the corresponding instance of an MCTP Request message.
Command Code	For Request messages, this field is a command code indicating the type of MCTP operation the packet is requesting. Command code values are defined in Table 12. The format and definition of request and response parameters for the commands is given in Clause 12. The Command Code that is sent in a Request shall be returned in the corresponding Response.

Field Name	Description
Completion Code	This field is only present in Response messages. This field contains a value that indicates whether the response completed normally. If the command did not complete normally, the value can provide additional information regarding the error condition. The values for completion codes are specified in Table 13.
Message Data	Zero or more bytes of parameter data that is specific to the particular Command Code and whether the message is a Request or Datagram, or a Response.
* These fields are MCTP base protocol fields.	

1770 **11.4 MCTP control message transmission unit size**

1771 All MCTP control messages are required to have a packet payload that is no larger than the baseline
1772 transmission unit size of 64 bytes.

1773 MCTP control messages are carried in a single MCTP packet. Multiple messages are used if an operation
1774 requires more data to be transferred than can be carried in a single message.

1775 **11.5 Tag Owner (TO), Request (Rq), and Datagram (D) bit usage**

1776 For MCTP control messages, the Rq bit shall be set to 1b if the message is a “command” or Request
1777 message and 0b if the message is a Response message. For Datagram and Broadcast messages, the
1778 Rq bit shall always be set to 1b. MCTP Control messages that have unexpected or incorrect flag bit
1779 values shall be silently discarded by the receiver of the message.

1780 For the present specification, Requests and Datagrams are only issued from tag owners (TO bit = 1b).
1781 Provision has been left for the definition of possible future Datagrams that are not issued from tag owners
1782 (see Table 11).

1783 **Table 11 – Tag Owner (TO), Request (Rq) and Datagram (D) bit usage**

MCTP Control Message Class	Destination EID Value	Tag Owner (TO) bit	Request (Rq) bit	Datagram (D) bit
Command/Request Responses are expected and tracked by Instance ID at the requester.	Target EID	1b	1b	0b
Response	Target EID	0b	0b	0b
Broadcast Request Responses are expected and tracked by Instance ID at the requester.	Broadcast EID	1b	1b	0b
Datagram Unacknowledged Request – Responses are neither expected nor tracked by Instance ID at the requester. Duplicate packets are handled the same as retried Command/Request packets.	Target EID	1b	1b	1b
Broadcast Datagram (unacknowledged control command that is broadcast.)	Broadcast EID	1b	1b	1b
Reserved for future definition	all other			

1784 11.6 Concurrent command processing

1785 This clause describes the specifications and requirements for handling concurrent overlapping MCTP
1786 control requests by endpoints.

1787 11.6.1 Requirements for responders

1788 An endpoint is not required to process more than one request at a time (that is, it can be “single threaded”
1789 and does not have to accept and act on new requests until it has finished responding to any previous
1790 request).

1791 A responder that is not ready to accept a new request can either silently discard the request, or it can
1792 respond with an `ERROR_NOT_READY` message completion code.

1793 A responder that can accept and process more than one request at a time is not required to return
1794 responses in the order that the requests were received.

1795 11.6.2 Requirements for Requesters

1796 An endpoint that issues MCTP control Requests to another endpoint shall wait until it gets the response
1797 to the particular request, or times out waiting for the response, before issuing a new request, Datagram,
1798 or Broadcast Datagram.

1799 An endpoint that issues MCTP control Requests is allowed to have multiple requests outstanding
1800 simultaneously to *different* responder endpoints.

1801 An endpoint that issues MCTP control Requests should be prepared to handle responses that may not
1802 match the request (that is, it should not automatically assume that a response that it receives is for a
1803 particular request). It should check to see that the command code and source EID values in the response
1804 match up with a corresponding outstanding command before acting on any parameters returned in the
1805 response.

1806 11.6.3 Additional requirements for bridges

1807 The packets that are routed *through* a bridge’s routing functionality are not interpreted by the bridge and
1808 therefore are not considered to constitute concurrent requests.

1809 A bridge shall support at least one outstanding MCTP control request for each bus connection (port)
1810 through which MCTP control messages can be used to access the bridge’s configuration and control
1811 functionality.

1812 Bridges shall retain temporal ordering of packets forwarded from one message terminus to another.

1813 12 MCTP control messages

1814 This clause contains detailed descriptions for each MCTP control message. The byte offsets for the
1815 Request and Response parameter information given in the tables for the commands indicates the byte
1816 offset for the message data starting with the byte following the Command field.

1817 12.1 MCTP control message command codes

1818 Table 12 lists the MCTP control messages and their corresponding command code values. The
1819 commands and their associated parameters are specified later in this clause. For bridges, the
1820 requirements apply equally to all endpoints within the bridge device that are used to configure and control
1821 the bridges routing functionality.

Table 12 – MCTP control command numbers

Command Code	Command Name	General Description	OMC		Clause
			E	B	
0x00	Reserved	Reserved	–	–	–
0x01	Set Endpoint ID	Assigns an EID to the endpoint at the given physical address	Ma Ng	Ca ¹ Mg	12.3
0x02	Get Endpoint ID	Returns the EID presently assigned to an endpoint. Also returns information about what type the endpoint is and its level of use of static EIDs.	Ma Og	Ma Og	12.4
0x03	Get Endpoint UUID	Retrieves a per-device unique UUID associated with the endpoint	Ca ² Og ⁹	Ca ² Og	12.5
0x04	Get MCTP Version Support	Lists which versions of the MCTP control protocol are supported on an endpoint	Ma Og	Ma Og ⁵	12.6
0x05	Get Message Type Support	Lists the message types that an endpoint supports	Ma Og	Ma Og	12.7
0x06	Get Vendor Defined Message Support	Used to discover an MCTP endpoint's vendor-specific MCTP extensions and capabilities	Oa Og	Oa Og	12.8
0x07	Resolve Endpoint ID	Used to get the physical address associated with a given EID	Na Og	Ma Og	12.9
0x08	Allocate Endpoint IDs	Used by the bus owner to allocate a pool of EIDs to an MCTP bridge	Na Ng	Ma ⁶ Mg ⁶	12.10
0x09	Routing Information Update	Used by the bus owner to extend or update the routing information that is maintained by an MCTP bridge	Oa ⁸ Og ⁸	Ma ⁴ Mg ⁴	12.11
0x0A	Get Routing Table Entries	Used to request an MCTP bridge to return data corresponding to its present routing table entries	Na Og	Ma Og	12.12
0x0B	Prepare for Endpoint Discovery	Used to direct endpoints to clear their "discovered" flags to enable them to respond to the Endpoint Discovery command	Ca ³ Ng	Ca ³ Cg ³	12.13
0x0C	Endpoint Discovery	Used to discover MCTP-capable devices on a bus, provided that another discovery mechanism is not defined for the particular physical medium	Ca ³ Cg ³	Ca ³ Cg ³	12.14
0x0D	Discovery Notify	Used to notify the bus owner that an MCTP device has become available on the bus	Na Cg ³	Ca ³ Cg ³	12.15
0x0E	Get Network ID	Used to get the MCTP network ID	Ca ⁷	Ca ⁷	12.16
0x0F	Query Hop	Used to discover what bridges, if any, are in the path to a given target endpoint and what transmission unit sizes the bridges will pass for a given message type when routing to the target endpoint	Na Og	Ma Og	12.17
0x10	Resolve UUID	Used by endpoints to find another endpoint matching an endpoint that uses a specific UUID.	Na Og	Oa Og	12.18
0x11	Query rate limit	Used to discover the data rate limit settings of the given target for incoming messages.	Oa Og	Oa Og	12.19
0x12	Request TX rate limit	Used to request the allowed transmit data rate limit for the given endpoint for outgoing messages.	Oa Og	Oa Og	12.20

Command Code	Command Name	General Description	OMC		Clause
			E	B	
0x13	Update rate limit	Used to update the receiving side on change to the transmit data rate which was not requested by the receiver	Oa Og	Oa Og	12.21
0x14	Query Supported Interfaces	Used to discover the existing device MCTP interfaces.	Oa Og	Oa Og	
0xF0 – 0xFF	Transport Specific	This range of control command numbers is reserved for definition by individual MCTP Transport binding specifications. Transport specific commands are intended to be used as needed for setup and configuration of MCTP on a given media. A particular transport specific command number may have different definitions depending on the binding specification. Transport specific commands shall only be addressed to endpoints on the same medium. A bridge is allowed to block transport specific commands from being bridged to different media. The general format of Transport specific messages is specified in clause 12.17.	-	-	12.23

Key for OMC (optional / mandatory / conditional) column:

- E = non-bridge, non-bus owner endpoint (simple endpoint)
- B = bridge / bus-owner endpoint
- Ma = mandatory (required) to accept. The request shall be accepted by the endpoint and a response generated per the following command descriptions.
- Mg = mandatory to generate. The endpoint shall generate this request as part of its responsibilities for MCTP operation.
- Oa = optional to accept
- Og = optional to generate
- Ca = conditional to accept (see notes)
- Cg = conditional to generate (see notes)
- Na = not applicable to accept. This command is not applicable to the device type and shall not be accepted
- Ng = not applicable to generate. This command is used for MCTP configuration and initialization and should not be generated.

1. The topmost bus owner is not required to support the Set Endpoint ID command.
2. Hot-plug and add-in devices, and non-bridge devices that connect to multiple busses, are required to support the Get Endpoint UUID command. See 8.17.7 and 8.17.8 for more info.
3. Mandatory on a per-bus basis to support endpoint discovery if required by the physical transport binding used for the particular bus type. Refer to the appropriate MCTP physical transport binding specification.
4. The topmost bus owner is not required to accept this command. The command is required to be generated when downstream bridges require dynamic routing information from bus owners that they are connected to. Some implementations may be configured where all routing information has been statically configured into the bridge and no dynamically provided information is required. In this case, it is not required to support the command while the endpoints are configured in that manner.
5. Bridges should use this command to verify that they are initializing devices that are compatible with their MCTP control protocol version.
6. The endpoint is required to accept this command if it indicated support for a dynamic EID pool. The command shall be generated by the endpoint if the configuration requires the endpoint to support allocating EID pools to downstream bridges.
7. See Clause 9 MCTP Network IDs for information for implementation requirements of this command.
8. While it is optional for an endpoint to receive a routing information update, the MCTP Base specification does not specify a bridge or bus owner function that sends such updates to particular endpoints.
9. While it is optional for an endpoint to support this command, support of this command is mandatory both to generate and to accept for devices which supporting rate limiting.

1823 **12.2 MCTP control message completion codes**

1824 The command/result code field is used to return management operation results for response messages. If
 1825 a `SUCCESS` completion code is returned then the specified response parameters (if any) shall also be
 1826 returned in the response. If an error completion code (not `SUCCESS`) is returned by the responder, unless
 1827 otherwise specified, the responder shall not return any additional parametric data and the requester shall
 1828 ignore any additional parameter data provided in the response (if any). See Table 13 for the completion
 1829 codes.

1830 **Table 13 – MCTP control message completion codes**

Value	Name	Description
0x00	SUCCESS	The Request was accepted and completed normally.
0x01	ERROR	This is a generic failure message. (It should not be used when a more specific result code applies.)
0x02	ERROR_INVALID_DATA	The packet payload contained invalid data or an illegal parameter value.
0x03	ERROR_INVALID_LENGTH	The message length was invalid. (The Message body was larger or smaller than expected for the particular request.)
0x04	ERROR_NOT_READY	The Receiver is in a transient state where it is not ready to receive the corresponding message.
0x05	ERROR_UNSUPPORTED_CMD	The command field in the control type of the received message is unspecified or not supported on this endpoint. This completion code shall be returned for any unsupported command values received in MCTP control Request messages.
0x80–0xFF	COMMAND_SPECIFIC	This range of completion code values is reserved for values that are specific to a particular MCTP control message. The particular values (if any) and their definition is provided in the specification for the particular command.
All other	Reserved	Reserved

1831 **12.3 Set Endpoint ID**

1832 The Set Endpoint ID command assigns an EID to an endpoint. This command should only be issued by a
 1833 bus owner to assign an EID to an endpoint at a particular physical address. Since it is assumed the
 1834 Endpoint does not already have an EID assigned to it, or because the EID is unknown, the destination
 1835 EID in the message will typically be set to the special null destination EID value.

1836 The Set Endpoint ID command is also used to provide the Physical Address and EID of the Bus Owner to
 1837 an Endpoint. An Endpoint that needs to communicate with the Bus Owner may capture the physical
 1838 address and EID that was used to deliver the Set Endpoint ID message.

1839 Note: Endpoints that are not the Bus Owner should not issue the Set Endpoint ID command because it can
 1840 cause the receiver of the message to capture incorrect information for the Bus Owner's address.

1841 An MCTP bridge may elect to have a single EID for its functionality, rather than using an EID for each port
 1842 (bus connection) that is connected to a different bus owner. See 9.1.2 for more information. In this case,
 1843 the bridge will accept its EID assignment from the “first” bus to deliver the Set Endpoint ID request to the
 1844 bridge.

1845 It is recognized that different internal processing delays within a bridge can cause the temporal ordering
 1846 of requests to be switched if overlapping requests are received over more than one bus. Therefore, which
 1847 request is accepted by an implementation is not necessarily tied to the request that is first received at the
 1848 bridge, but instead will be based on which request is the first to be processed by the bridge.

1849 If an EID has already been assigned and the Set Endpoint ID command is issued from a different bus
 1850 without forcing an EID assignment, the command shall return a `SUCCESSFUL` completion code, but the
 1851 response parameters shall return an EID assignment status of “EID rejected”.

1852 The Set Endpoint ID command functions in the same manner regardless of whether the endpoint uses a
 1853 static EID. The only difference is that if an endpoint has a static EID, it uses that EID as its initial “default”
 1854 EID value. The endpoint does not treat this initial EID as if it were assigned to it by a different bus owner.
 1855 That is, the endpoint shall accept the EID assignment from the first bus that the command is received
 1856 from, and shall track that bus as the originating bus for the EID for subsequent instances of Set Endpoint
 1857 ID command. See 8.17.2 for more information. The request and response parameters are specified in
 1858 Table 14.

1859 **Table 14 – Set Endpoint ID message**

	Byte	Description
Request data	1	<p>Operation</p> <p>[7:3] – reserved</p> <p>[1:0] – Operation:</p> <p>00b Set EID. Submit an EID for assignment. The given EID will be accepted conditional upon which bus the device received the EID from (see preceding text). A device where the endpoint is only reached through one bus shall always accept this operation (provided the EID value is legal).</p> <p>01b Force EID. Force EID assignment. The given EID will be accepted regardless of whether the EID was already assigned through another bus. Note that if the endpoint is forcing, the EID assignment changes which bus is being tracked as the originator of the Set Endpoint ID command. A device where the endpoint is only reached through one bus shall always accept this operation (provided the EID value is legal), in which case the Set EID and Force EID operations are equivalent.</p> <p>10b Reset EID (optional). This option only applies to endpoints that support static EIDs. If static EIDs are supported, the endpoint shall restore the EID to the statically configured EID value. The EID value in byte 2 shall be ignored. An <code>ERROR_INVALID_DATA</code> completion code shall be returned if this operation is not supported.</p> <p>11b Set Discovered Flag. Set Discovered flag to the “discovered” state only. Do not change present EID setting. The EID value in byte 2 shall be ignored.</p> <p>Note that Discovered flag is only used for some physical transport bindings. An <code>ERROR_INVALID_DATA</code> completion code shall be returned if this operation is selected and the particular transport binding does not support a Discovered flag.</p>
	2	<p>Endpoint ID.</p> <p>0xFF, 0x00 = illegal.</p> <p>Endpoints are not allowed to be assigned the broadcast or null EIDs. It is recommended that the endpoint return an <code>ERROR_INVALID_DATA</code> completion code if it receives either of these values.</p>

	Byte	Description
Response data	1	Completion code
	2	[7:6] – reserved [5:4] – EID assignment status: 00b = EID assignment accepted. 01b = EID assignment rejected. EID has already been assigned by another bus owner and assignment was not forced. 10b = reserved. 11b = reserved. [3:2] – reserved. [1:0] – Endpoint ID allocation status (see 12.10 for additional information): 00b = Device does not use an EID pool. 01b = Endpoint requires EID pool allocation. 10b = Endpoint uses an EID pool and has already received an allocation for that pool. 11b = reserved
	3	EID Setting. If the EID setting was accepted, this value will match the EID passed in the request. Otherwise, this value returns the present EID setting.
	4	EID Pool Size. This is the size of the dynamic EID pool that the bridge can use to assign EIDs or EID pools to other endpoints or bridges. It does not include the count of any additional static EIDs that the bridge may maintain. See 8.17.2 for more information. Note that a bridge always returns its pool size regardless of whether it has already received an allocation. 0x00 = no dynamic EID pool.

1860 **12.4 Get Endpoint ID**

1861 The Get Endpoint ID command returns the EID for an endpoint. This command is typically issued only by
 1862 a bus owner to retrieve the EID that was assigned to a particular physical address. Thus, the destination
 1863 EID in the message will typically be set to the special Physical Addressing Only EID value. The request
 1864 and response parameters are specified in Table 15.

1865 **Table 15 – Get Endpoint ID message**

	Byte	Description
Request data	–	–
Response data	1	Completion Code.
	2	Endpoint ID. 0x00 = EID not yet assigned.
	3	Endpoint Type. [7:6] = reserved [5:4] = Endpoint Type: 00b = simple endpoint 01b = bus owner/bridge 10b = reserved

	Byte	Description
		<p>11b = reserved</p> <p>[2:0] = reserved</p> <p>[1:0] = Endpoint ID Type:</p> <p>00b = dynamic EID.</p> <p>The endpoint uses a dynamic EID only.</p> <p>01b = static EID supported.</p> <p>The endpoint was configured with a static EID. The EID returned by this command reflects the present setting and may or may not match the static EID value.</p> <p>The following two status return values are optional. If provided, they shall be supported as a pair in place of the static EID support status return. It is recommended that this be implemented if the Reset EID option in the Set Endpoint ID command is supported.</p> <p>10b = static EID supported.</p> <p>Present EID matches static EID.</p> <p>The endpoint has been configured with a static EID. The present value is the same as the static value.</p> <p>11b = static EID supported. Present EID does not match static EID.</p> <p>Endpoint has been configured with a static EID. The present value is different than the static value.</p> <p>See 8.17.2 for more information.</p>
	4	<p>Medium-Specific Information.</p> <p>This byte can hold additional information about optional configuration of the endpoint on the given medium, such as whether certain types of timing or arbitration are supported. This should only be used to report static information.</p> <p>This byte shall be returned as 0x00 unless otherwise specified by the transport binding.</p>

1866 **12.5 Get Endpoint UUID**

1867 The Get Endpoint UUID command returns a universally unique identifier (UUID), also referred to as a
 1868 globally unique ID (GUID), for the management controller or management device. The command can be
 1869 used to correlate a device with one or more EIDs. The format of the ID follows the byte (octet) format
 1870 specified in [RFC4122](#). [RFC4122](#) specifies four different versions of UUID formats and generation
 1871 algorithms suitable for use for a device UUID in IPMI. These are version 1 (0001b) “time based”, and
 1872 three “name-based” versions: version 3 (0011b) “MD5 hash”, version 4 (0100b) “Pseudo-random”, and
 1873 version 5 “SHA1 hash”. The version 1 format is recommended. However, versions 3, 4, or 5 formats are
 1874 also allowed. A device UUID should never change over the lifetime of the device. The request and
 1875 response parameters are specified in Table 16.

1876 See 8.17.7 and 8.17.8 for additional requirements on the use of the Get Endpoint UUID command.

1877 **Table 16 – Get Endpoint UUID message format**

	Byte	Description
Request data	–	–
Response data	1	Completion Code
	2:17	UUID bytes 1:16, respectively (see Table 17)

1878 The individual fields within the UUID are stored most-significant byte (MSB) first per the convention
 1879 described in [RFC4122](#). See Table 17 for an example format.

1880 **Table 17 – Example UUID format**

Field	UUID Byte	MSB
time low	1	MSB
	2	
	3	
	4	
time mid	5	MSB
	6	
time high and version	7	MSB
	8	
clock seq and reserved	9	MSB
	10	
node	11	MSB
	12	
	13	
	14	
	15	
	16	

1881 **12.6 Get MCTP version support**

1882 This command can be used to retrieve the MCTP base specification versions that the endpoint supports,
 1883 and also the message type specification versions supported for each message type. The format of the
 1884 request and response parameters for this message is given in Table 18.

1885 More than one version number can be returned for a given message type by the Get MCTP Version
 1886 Support command. This enables the command to be used for reporting different levels of compatibility
 1887 and backward compatibility with different specification versions. The individual specifications for the given
 1888 message type define the requirements for which versions number values should be used for that
 1889 message type. Those documents define which earlier version numbers, if any, shall also be listed.

1890 The command returns a completion code that indicates whether the message type number passed in the
 1891 request is supported or not. This enables the command to also be used to query the endpoint for whether
 1892 it supports a given message type.

1893 **NOTE** Version numbers are listed from oldest to newest. Versioning commands and version formats for vendor-
 1894 defined message types, 0x7E and 0x7F, are vendor-specific and considered outside the scope of this specification.

1895 **Table 18 – Get MCTP version support message**

	Byte	Description
Request data	1	Message Type Number The Message Type Number to retrieve version information for: 0xFF = return MCTP base specification version information.

	Byte	Description
		<p>0x7E, 0x7F = unspecified. Support of this command for vendor-defined message types is vendor implementation-specific and considered outside the scope of this specification.</p> <p>0x00 = return MCTP control protocol message version information.</p> <p>0x01 = return version of DSP0241</p> <p>0x02,0x03 = return version of DSP0261</p> <p>Other = return version information for a given message type. See MCTP ID for message type numbers. When a Message Type Number references a binding spec, the reported version is of the binding spec and not of the associated base spec.</p>
Response data	1	<p>Completion Code</p> <p>0x80 = message type number not supported</p>
	2	<p>Version Number Entry count</p> <p>One-based count of 32-bit version numbers being returned in this response. Numerically lower version numbers are returned first.</p>
	3:6	<p>Version Number entry 1: The following descriptions are informational. Refer to DSP4004 for the normative definition of version numbering of DMTF specifications.</p> <p>[31:24] = major version number. This field is used to identify a version of the specification that includes changes that make it incompatible with one or more functions that were defined in versions of the specification that have an older (smaller) major version number.</p> <p>[23:16] = minor version number. This field is used to identify functional additions to the specification that are backward compatible with older (smaller) minor version numbers that share the same major version number.</p> <p>[15:8] = update version number. This field is used for editorial updates to the specification that do not define new functionality nor change existing functionality over the given major.minor release. This field is informational and should be ignored when checking versions for interoperability.</p> <p>[7:0] = "alpha" byte. This value is used for pre-release (work-in-progress) versions of the specification. Pre-release versions of the specification are backward compatible with specification versions that have an older (smaller) minor version numbers that share the same major version number. However, since the alpha value represents a version of the specification that is presently under development, versions that share the same major and minor version numbers, but have different 'alpha' versions may not be fully interoperable.</p> <p>The encoding of the version number and alpha fields is provided in 12.6.1.</p>
	(7:X)	<p>Version Number Entries 2 through N.</p> <p>Additional 32-bit major/minor version numbers, if any.</p>

1896 12.6.1 Version field encoding

1897 The version field is comprised of four bytes referred to as the "major", "minor", "update", and "alpha"
 1898 bytes. These bytes shall be encoded as follows:

1899 The "major", "minor", and "update" bytes are BCD-encoded, and each byte holds two BCD digits. The
 1900 "alpha" byte holds an optional alphanumeric character extension that is encoded using one of the

1901 alphabetic characters [a-z, A-Z] from the US-ASCII ([RFC20](#)) Character Set. The semantics of these fields
1902 follows that specified in [DSP4004](#).

1903 The value 0x00 in the alpha field means that the alpha field is not used. Software or utilities that display
1904 the version number should not display any characters for this field.

1905 The value 0xF in the most-significant nibble of a BCD-encoded value indicates that the most-significant
1906 nibble should be ignored and the overall field treated as a single-digit value. Software or utilities that
1907 display the number should only display a single digit and should not put in a leading "0" when displaying
1908 the number.

1909 A value of 0xFF in the "update" field indicates that the field to be ignored. Software or utilities that display
1910 the version number should not display any characters for the field. 0xFF is not allowed as a value for the
1911 "major" or "minor" fields.

1912 EXAMPLES:

1913 Version 1.1.0 → 0xF1F1F000

1914 Version 3.1 → 0xF3F1FF00

1915 Version 1.0a → 0xF1F0FF61

1916 Version 3.7.10a → 0xF3F71061

1917 Version 10.11.7 → 0x1011F700

1918 **12.6.2 MCTP base specification version number**

1919 MCTP implementations that follow this particular specification shall return the following version information in the
1920 response to the Get MCTP Version Support message when the Message Type parameter in the request is set to
1921 0xFF (return MCTP base specification version information).

1922 The Version Number Entry 1 field shall be used to indicate backward compatibility with Version 1.0 of the
1923 base specification as:

1924 **36.2**[Major version 1, minor version 0, no update version, no alpha]

1925 This is reported using the encoding as: 0xF1F0FF00

1926 The Version Number Entry 2 field shall be used to indicate backward compatibility with Version 1.1 of the
1927 base specification as:

1928 **36.2.0** [Major version 1, minor version 1, update version 0, no alpha]

1929 This is reported using the encoding as: 0xF1F1F000

1930 The version of the MCTP base specification for this specification shall be reported in Version Number
1931 Entry 3 as:

1932 **1.3.0** [Major version 1, minor version 3, update version 0, no alpha]

1933 This is reported using the encoding as: 0xF1F2F000

1934 **12.6.3 MCTP control protocol version information**

1935 MCTP implementations that follow this particular specification shall return the following version information in the
1936 response to the Get MCTP Version Support message when the Message Type parameter in the request is set to
1937 0x00 (return MCTP control protocol version information).

1938 The Version Number Entry 1 field shall be used to indicate backward compatibility with Version 1.0 of the
1939 base specification Control Protocol as:

1940 **36.2**[Major version 1, minor version 0, no update version, no alpha]

1941 This is reported using the encoding as: 0xF1F0FF00

1942 The Version Number Entry 2 field shall be used to indicate backward compatibility with Version 1.1 of the
1943 base specification Control Protocol as:

1944 **36.2.0** [Major version 1, minor version 1, update version 0, no alpha]

1945 This is reported using the encoding as: 0xF1F1F000

1946 The version of the MCTP base specification Control Protocol for this specification shall be reported in
1947 Version Number Entry 3 as:

1948 **1.2.0** [Major version 1, minor version 2, update version 0, no alpha]

1949 This is reported using the encoding as: 0xF1F2F000

1950 **12.7 Get Message Type Support**

1951 The Get Message Type Support command enables management controllers to discover the MCTP
1952 control protocol capabilities supported by other MCTP endpoints, and get a list of the MCTP message
1953 types that are supported by the endpoint. The request and response parameters for this message are
1954 listed in Table 19.

1955 The response to this command may be specific according to which bus the request was received over
1956 (that is, a device that supports a given message type may not support that message type equally across
1957 all buses that connect to the device).

1958 **Table 19 – Get Message Type Support message**

	Byte	Description
Request data	–	–
Response data	1	Completion Code.
	2	MCTP Message Type Count. One-based. Number of message types in addition to the MCTP control message type that is supported by this endpoint
	(3:N)	List of Message Type numbers. One byte per number. See Table 3 and MCTP.ID .

1959 **12.8 Get Vendor Defined Message Support**

1960 The Get Vendor Defined Message Support operation enables management controllers to discover
1961 whether the endpoint supports vendor-defined messages, and, if so, the vendors or organizations that
1962 defined those messages. The format and definition of the request and response parameters for this
1963 message is given in Table 20.
1964

1965

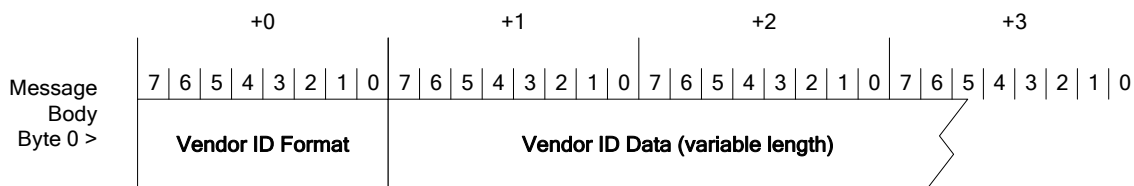
Table 20 – Get Vendor Defined Message Support message

	Byte	Description
Request data	1	Vendor ID Set Selector Indicates the specific capability set requested. Indices start at 0x00 and increase monotonically by 1. If the responding endpoint has one or more capability sets with indices greater than the requested index, it increments the requested index by 1 and returns the resulting value in the response message. The requesting endpoint uses the returned value to request the next capability set.
Response data	1	Completion Code
	2	Vendor ID Set Selector 0xFF = no more capability sets.
	Var	Vendor ID A structured field of variable length that identifies the vendor ID format (presently PCI or IANA) and the ID of the vendor that defined the capability set. The structure of this field is specified in Figure 21 – Structure of Vendor ID field for Get Vendor Defined capabilities message.
	2 bytes	16-bit numeric value or bit field, as specified by the vendor or organization identified by the vendor ID. This value is typically used to identify a particular command set type or major version under the given vendor ID.

1966 **12.8.1 Vendor ID formats**

1967 Figure 21 shows the general structure of Vendor ID fields used in this specification. The first byte of the
 1968 field contains the Vendor ID Format, a numeric value that indicates the definition space and format of the
 1969 ID. The remainder of the field holds the Vendor ID Data with content and format as specified in Table 21.

1970 The MCTP management controller or management device can pick which format is best suited for the
 1971 device. In general, if the device does not already have an existing vendor ID that matches one of the
 1972 specified formats, it is recommended that the IANA enterprise number format be used.



1973

1974 **Figure 21 – Structure of Vendor ID field for Get Vendor Defined capabilities message**

1975

Table 21 – Vendor ID formats

Vendor ID Format Name	Vendor ID Format	Vendor ID Data Length	Description
PCI Vendor ID	0x00	2	16-bit Unsigned Integer. The PCI 2.3 specifications state the following about the PCI vendor ID: “This field identifies the manufacturer of the device. Valid vendor identifiers are allocated by the PCI SIG to ensure uniqueness. 0xFFFF is an invalid value for the Vendor ID.” However, for MCTP this value may be used for identifying aspects other than the manufacturer of the device, such as its use in the Vendor Defined – PCI message type, where it identifies the vendor or organization that defined a particular set of vendor-defined messages. Thus, in some uses, the ID may or may not correspond to the PCI ID for the manufacturer of the device.
IANA Enterprise Number	0x01	4	32-bit Unsigned Integer. The IANA enterprise number for the organization or vendor expressed as a 32-bit unsigned binary number. For example, the enterprise ID for the DMTF is 412 (decimal) or 0x0000_019C expressed as a 32-bit hexadecimal number. The enterprise number is assigned and maintained by the Internet Assigned Numbers Authority, www.iana.org, as a means of identifying a particular vendor, company, or organization.

1976 12.9 Resolve Endpoint ID

1977 This command is sent to the bus owner to resolve an EID into the physical address that shall be used to
 1978 deliver MCTP messages to the target endpoint. The command takes an EID as an input parameter in the
 1979 request and returns the EID and the physical address for routing to that EID (if any) in the response. The
 1980 response data will also indicate if no mapping was available.

1981 An endpoint knows the physical address of the bus owner by keeping track of which physical address
 1982 was used when the endpoint received its EID assignment through the Set Endpoint ID command. The
 1983 endpoint can send this command to the bus owner using the null destination EID value. This eliminates
 1984 the need for the endpoint to also keep track of the EID of the bus owner. The request and response
 1985 parameters are specified in Table 22.

1986

Table 22 – Resolve Endpoint ID message

	Byte	Description
Request data	1	Target Endpoint ID This is the EID that the bus owner is being asked to resolve.
Response data	1	Completion Code
	2	Bridge Endpoint ID This is the EID for the endpoint that is providing the bridging server (if any) that is required to access the target endpoint. If the EID being returned matches the same value as the target EID, it indicates that there is no bridging function that is required to access the target endpoint (that is, the target EID is local to the bus that the Resolve Endpoint ID request was issued over).
	3:N	Physical Address.

	Byte	Description
		The size of this field is dependent on the particular MCTP physical transport binding used for the bus that this data is being provided for. The size and format of this field is defined as part of the corresponding physical transport binding specification.

1987 12.10 Allocate Endpoint IDs

1988 Bus owners are responsible for allocating pools of EIDs to MCTP bridges that are lower in the bus
 1989 hierarchy. This is done using the Allocate Endpoint IDs command. The EID for the bridge itself is
 1990 assigned separately and is *not* part of the pool given with this command.

1991 The bus owner will typically use this command as part of the EID assignment process for a bus. When a
 1992 device has been assigned an EID using the Set Endpoint ID command, the response to that command
 1993 indicates whether the endpoint supports an EID pool. If the device indicates that it supports an EID pool,
 1994 the bus owner can then issue the Allocate Endpoint IDs command to supply the pool of EIDs to the
 1995 device.

1996 NOTE: The Allocate Endpoint IDs command can also cause a bridge to rebuild its routing table. See 12.11.2 for
 1997 more information.

1998 When an EID or EID pool that was previously allocated becomes unused (for example, due to a hot-swap
 1999 removal), the bus owner shall reclaim the endpoint's EID or EID pool allocation. See 8.17 for additional
 2000 details.

2001 Referring to Figure 22, there is a potential race condition with handling EID allocation. In the scenario
 2002 shown in this figure, it is possible that device X and device Z might both be assigning EIDs to device Y at
 2003 the same time. This also means that, unless steps are taken, device Z could allocate endpoints to device
 2004 Y only to have this overwritten by a set of endpoints assigned by device X.

2005 To prevent this, the Allocate Endpoint IDs command is only accepted from the "first" bus that provides the
 2006 EID pool to the device. If another bus owner attempts to deliver an EID pool through another bus, the
 2007 request will be rejected unless an intentional over-ride is done.

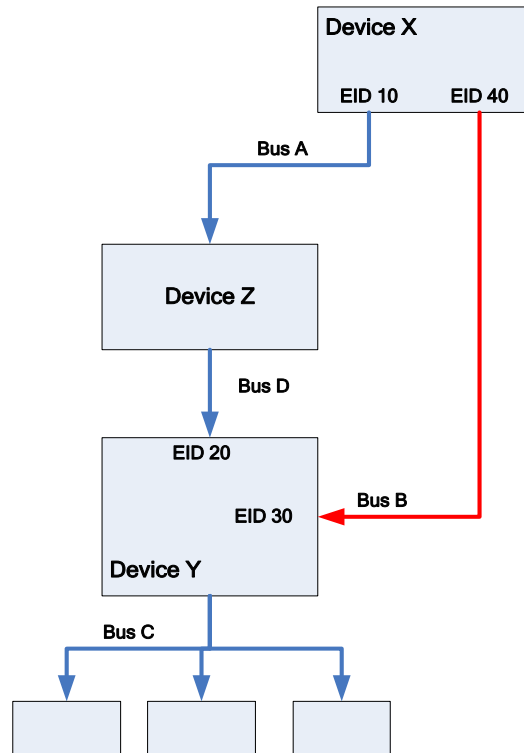


Figure 22 – EID Pools from multiple bus owners

2008

2009

2010 The Allocate Endpoint IDs message fields are described in Table 23.

2011

Table 23 – Allocate Endpoint IDs message

	Byte	Description
Request data	1	Operation Flags: [7:1] – reserved. [1:0] – Operation: 00b = Allocate EIDs. Submit an EID pool allocation. Do not force allocation. This enables the allocation to be rejected if the bridge has already received its EID pool from another bus. (See additional information in the following clauses.) 01b = Force allocation. Force bridge to accept this EID pool regardless of whether it has already received its EID pool from another bus. This shall also cause a bridge to rebuild its routing tables. See 12.11.2 for more information. 10b = Get allocation information Return the response parameters without changing the present allocation. This can be used to query information on the dynamic pool of EIDs presently allocated to the Endpoint, if any. If this operation is selected, the Number of Endpoint IDs and Starting Endpoint ID parameters in the request shall be ignored. 11b = Reserved
	2	Number of Endpoint IDs (Allocated Pool Size) Specifies the number of EIDs in the pool being made available to this Endpoint Specifying a count of 0x00 shall be legal. If 0x00 is accepted or forced (and

	Byte	Description
		the bridge lacks a static EID pool) no EIDs shall be available for distribution by the particular bridge.
	3	Starting Endpoint ID Specifies the starting EID for the range of EIDs being allocated in the pool. When multiple EIDs are provided, the IDs are sequential starting with this value as the first EID in the range.
Response data	1	Completion Code An error completion code (ERROR_INVALID_DATA should be returned) shall be returned if the number of EIDs being allocated (Number of Endpoint IDs) exceeds the Dynamic Endpoint ID Pool size. (This error condition does not apply to when the number of endpoint IDs passed in the request is 0x00).
	2	[7:2] – reserved [1:0] – 00b = Allocation was accepted. In the case that the bridge has a completely static EID pool, the bridge should not track which bus has sourced the command and shall accept the allocation if the Number of Endpoint IDs (Allocated Pool Size) is 0x00. 01b = Allocation was rejected. The Allocate Endpoint IDs command is accepted only from the “first” bus that provides the EID pool to the device. If another bus owner attempts to deliver an EID pool through another bus, the request will be rejected unless an intentional over-ride is done. (The rationale for this behavior is explained in the text of this clause.) 10b, 11b = reserved
	3	Endpoint ID Pool Size (Dynamic) This value is the size of the EID pool used by this endpoint. This is the size of the dynamic EID pool that the bridge can use to assign EIDs or EID pools to other endpoints or bridges. It does not include the count of any additional static EIDs that the bridge may maintain. See 8.17.2 for more information.
	4	First Endpoint ID This field specifies the first EID assigned to the pool for this endpoint. The value is 0x00 if there are no EIDs assigned to the pool.

2012 **12.11 Routing Information Update**

2013 The Routing Information Update message is used by a bus owner to give routing information to a bridge
2014 for the bus on which the message is being received.

2015 Because the physical address format is based on the bus over which the request is delivered, the bus
2016 owner shall use the medium-specific physical address format for the addresses sent using this command.

2017 An MCTP bridge may be sent more than one instance of this command to transfer the update information.

2018 An integral number of routing information update entries shall be provided in the command (that is,
2019 routing information update entries cannot be split across instances of the command).

2020 **12.11.1 Adding and replacing entries**

2021 The recipient of this command shall check to see whether the information in the request corresponds to
 2022 the EID for an existing entry for the bus over which the command was received. If so, it shall replace that
 2023 entry with the new information. If an entry for a given EID or EID range does not already exist, it shall
 2024 create new entries for the given EIDs. In some cases this may require the bridge to split existing entries
 2025 into multiple entries.

2026 NOTE: A bus owner is only allowed to update entries that correspond to its bus. For each routing table entry that
 2027 was created or updated through the Routing Information Update message, the bridge shall keep track of which bus it
 2028 received the Routing Information Update from. This is necessary so that when a Routing Information Update is
 2029 received from a particular bus, the bridge only updates entries that correspond to entries that were originally given to
 2030 it from that bus.

2031 **12.11.2 Rebuilding routing tables**

2032 A bridge that receives and accepts the Allocate Endpoint IDs command with the “Force Allocation” bit set
 2033 (1b) shall clear out and rebuild its routing table information. The bridge shall issue commands to reassign
 2034 EIDs and re-allocate EID pools to all downstream devices. The request and response parameters are
 2035 specified in Table 24, and format information is provided in Table 25.

2036 **Table 24 – Routing Information Update message**

	Byte	Description
Request data	1	Count of update entries (1-based)
	see text	One or more update entries, based on the given count, as illustrated in Table 25
Response data	1	Completion Code 0x80 = Insufficient space to add requested entries to internal routing table

2037 **Table 25 – Routing Information Update entry format**

Byte	Description
1	[7:4] – reserved [3:0] – Entry Type: 00b = entry corresponds to a single endpoint that is not serving 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 is serving as an MCTP bridge 11b = entry is an EID range for a bridge, but does not include the EID of the bridge itself
2	[7:0] Size of EID Range. The count of EIDs in the range.
3	First EID in EID Range. The EID Range is sequential (for example, if the size of the EID Range is 3 and the First EID value given in this parameter is 21, the Entry covers EIDs 21, 22, and 23).
4:N	Physical Address. The size and format of this field is defined as part of the corresponding physical transport binding specification for the bus that this data is being provided for.

2038 **12.12 Get Routing Table Entries**

2039 This command can be used to request an MCTP bridge or bus owner to return data corresponding to its
 2040 present routing table entries. This data is used to enable troubleshooting the configuration of routing
 2041 tables and to enable software to draw a logical picture of the MCTP network. More than one instance of
 2042 this command will typically need to be issued to transfer the entire routing table content.

2043 An integral number of routing table entries shall be provided in the response to this command (that is,
 2044 routing table entries cannot be split across instances of the command). The request and response
 2045 parameters are specified in Table 26, and format information is provided in Table 27.

2046 **Table 26 – Get Routing Table Entries message**

	Byte	Description
Request data	1	Entry Handle (0x00 to access first entries in table)
Response data	1	Completion Code
	2	Next Entry Handle (Use this value to request the next set of entries, if any.) If the routing table data exceeds what can be carried in a single MCTP control response. 0xFF = No more entries
	3	Number of routing table entries being returned in this response
	4:N	One or more routing table entries, formatted per Table 27. This field will be absent if the number of routing table entries is 0x00.

2047 **Table 27 – 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 reporting entry information using this command. While an implementation may internally organize its data that way, dynamic and statically configured routing shall 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 that the physical address for the entry is defined under. In some cases, this number

Byte	Description
	<p>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 or recommendations on how port numbers are assigned to corresponding physical connections on a device.</p>
4	Physical Transport Binding Identifier, according to DSP0239.
5	Physical Media Type Identifier, according to DSP0239. This value is used to indicate what format the following physical address data is given in.
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.

2048 12.13 Prepare for Endpoint Discovery

2049 The Endpoint Discovery message is used to determine if devices on a bus communicate MCTP (see
2050 Table 28). Whether this message is required depends on the particular medium. Currently, this message
2051 may be required only by a particular transport binding, such as PCI Express (PCIe) VDM, because other
2052 bindings such as SMBus/I2C may use other mechanisms for determining this information.

2053 Each endpoint (except the bus owner) on the bus maintains an internal flag called the “Discovered” flag.

2054 The Prepare for Endpoint Discovery command is issued as a broadcast Request message on a given bus
2055 that causes each endpoint on the bus to set their respective Discovered flag to the “undiscovered” state.
2056 The flag is subsequently set to the “discovered” state when the Set Endpoint ID command is received by
2057 the endpoint.

2058 An endpoint also sets the flag to the “undiscovered” state at the following times:

- 2059 • Whenever the physical address associated with the endpoint changes or is assigned
- 2060 • Whenever an endpoint first appears on the bus and requires an EID assignment
- 2061 • During operation if an endpoint enters a state that requires its EID to be reassigned
- 2062 • For hot-plug endpoints: After exiting any temporary state where the hot-plug endpoint was
2063 unable to respond to MCTP control requests for more than $T_{RECLAIM}$ seconds (where $T_{RECLAIM}$ is
2064 specified in the physical transport binding specification for the medium used to access the
2065 endpoint). See 8.17.5 for additional information.

2066 Only endpoints that have their Discovered flag set to “undiscovered” will respond to the Endpoint
2067 Discovery message. Endpoints that have the flag set to “discovered” will not respond.

2068 The destination EID for the Prepare for Endpoint Discovery message is set to the Broadcast EID value
2069 (see Table 2) in the request message to indicate that this is a broadcast message. The response
2070 message sets the destination EID to be the ID of the source of the request message, which is typically the
2071 EID of the bus owner. The request and response parameters are specified in Table 28.

2072 The Prepare for Endpoint Discovery message has no effect on existing EID assignments. That is,
2073 endpoints shall normally retain their EIDs until they are explicitly changed via the Set Endpoint ID
2074 command, and shall not clear them after getting a “Prepare for Endpoint Discovery” command. (Note that

2075 endpoints may lose their EIDs under other conditions such as power state changes, etc., as described
 2076 elsewhere in this specification.)

2077 The Endpoint Discovery and Prepare for Endpoint Discovery commands may only be supported on
 2078 particular transport bindings (e.g. MCTP over PCIe Vendor Defined Messaging). If the binding does not
 2079 use this discovery approach (e.g. SMBus/I2C) the endpoint shall return an `ERROR_UNSUPPORTED_CMD`
 2080 completion status for those commands.

2081 **Table 28 – Prepare for Endpoint Discovery message**

	Byte	Description
Request data	–	–
Response data	1	Completion Code

2082 **12.14 Endpoint Discovery**

2083 This command is used to discover endpoints that have their Discovered flag set to “undiscovered”. Only
 2084 endpoints that have their Discovered flag set to “undiscovered” will respond to this message. Endpoints
 2085 that have the flag set to “discovered” will not respond.

2086 This message is typically sent as a Broadcast Request message by the bus owner using the Broadcast
 2087 EID as the destination EID, though for testing purposes endpoints shall also accept and handle this
 2088 command as a non-broadcast Request. Additionally, the request may be sent as a datagram, depending
 2089 on the transport binding requirements. The request and response (if any) parameters are specified in
 2090 Table 29.

2091 **Table 29 – Endpoint Discovery message**

	Byte	Description
Request data	–	–
Response data	1	Completion Code

2092 **12.15 Discovery Notify**

2093 This message is available for use as a common message for enabling an endpoint to announce its
 2094 presence to the bus owner. This will typically be used as part of the endpoint discovery process when an
 2095 MCTP device is hot-plugged onto or becomes powered up on an MCTP bus.

2096 Whether and how this message is used for endpoint discovery depends on the particular physical
 2097 transport binding specification. For example, the SMBus/I2C transport binding does not use this message
 2098 for an endpoint to announce itself because it takes advantage of mechanisms that are already defined for
 2099 SMBus.

2100 This message should only be sent from endpoints to the bus owner for the bus that the endpoint is on so
 2101 it can notify the bus owner that the endpoint has come online and may require an EID assignment or
 2102 update. Additionally, the request may be sent as a datagram, depending on the transport binding
 2103 requirements. The request and response (if any) parameters are specified in Table 30.

2104 **Table 30 – Discovery Notify message**

	Byte	Description
Request data	–	–
Response data	1	Completion Code

2105 **12.16 Get Network ID**

2106 The Get Network ID command returns a universally unique identifier (UUID), also referred to as a globally
 2107 unique ID (GUID), for a given MCTP network. Typically this command is sent to the topmost MCTP bus-
 2108 owner since the topmost bus-owner has this knowledge. A Network ID is required for add-in MCTP
 2109 networks (For example, an MCTP Network on an add-in card or module). A Network ID is not required for
 2110 a fixed (not add-in) MCTP network provided there is only one network in the system implementation. A
 2111 Network ID is required for fixed MCTP networks when more than one fixed network exists in the system
 2112 implementation and is simultaneously accessible by a common entity such as system software.

2113 The format of the ID follows the byte (octet) format specified in [RFC4122](#). [RFC4122](#) specifies four
 2114 different versions of UUID formats and generation algorithms suitable for use for a device UUID in IPMI.
 2115 These are version 1 (0001b) “time based”, and three “name-based” versions: version 3 (0011b) “MD5
 2116 hash”, version 4 (0100b) “Pseudo-random”, and version 5 “SHA1 hash”. The version 1 format is
 2117 recommended. However, versions 3, 4, or 5 formats are also allowed. A device UUID should never
 2118 change over the lifetime of the device. The request and response parameters are specified in Table 16.

2119 **Table 31 – Get Network ID message format**

	Byte	Description
Request data	–	–
Response data	1	Completion Code
	2:17	Network ID bytes 1:16, respectively (see Table 17)

2120 The individual fields within the UUID are stored most-significant byte (MSB) first per the convention
 2121 described in [RFC4122](#). See Table 17 for an example format.

2122 **12.17 Query Hop**

2123 This command can be used to query a bridge to find out whether a given EID shall be accessed by going
 2124 through that bridge, and if so, whether yet another bridge shall be passed through in the path to the
 2125 endpoint, or if the endpoint is on a bus that is directly connected to the bridge.

2126 The command also returns the information about the transmission unit information that the bridge
 2127 supports in routing to the given target endpoint from the bus that the request was received over. See
 2128 clause 9.5 for more information.

2129 **NOTE** The physical transport binding for MCTP may place additional requirements on the physical packet sizes
 2130 that can be used to transfer MCTP packet payloads, such as requiring that physical packet sizes be in 32-byte or 64-
 2131 byte increments, or particular power of 2 increments (for example, 128, 256, 512, and so on).

2132 The request and response parameters are specified in Table 32.

2133 **Table 32 – Query Hop message**

	Byte	Description
Request data	1	Target Endpoint ID 0x00, 0xFF = reserved. (An <code>ERROR_INVALID_DATA</code> completion code shall be returned.)
	2	Message type for which transmission unit information is being requested. Use the MCTP control message type number unless another message type is of interest.

	Byte	Description
Response data	1	Completion Code An <code>ERROR_INVALID_DATA</code> completion code shall be returned if the target EID is not covered by any entry in the bridge's routing table.
	2	EID of the next bridge that is used to access the target endpoint, if any Note: This response depends on which bus port the Query Hop request is received over. If this EID is 00h: The EID is covered by the bridge's routing table, but the target EID does not require access by <i>going through</i> this bridge from the port the request was received over. This response will be returned if the target EID is already local to the bus over which the request is being received. This response is also returned when the target EID is an EID for the bridge itself. If this EID is non-zero <i>and</i> is different than the target EID passed in request: The EID being provided is the EID of the "next bridge" in the path to the target EID. If this EID is equal to the target EID passed in request: The target EID is accessed by going through this bridge and no additional bridges shall be gone through to reach the target.
	3	Message Type. This value either returns the message type that was given in the request, or it returns <code>0xFF</code> to indicate that the information is applicable to all message types that are supported by the bridge.
	4:5	Maximum supported incoming transmission unit size in increments of 16 bytes, starting from the baseline transmission unit size (<code>0x0000</code> = 64 bytes, <code>0x0001</code> = 80 bytes, and so on).
	5:6	Maximum supported outgoing transmission unit size in increments of 16 bytes, starting from the baseline transmission unit (<code>0x0000</code> = 64 bytes, <code>0x0001</code> = 80 bytes, and so on). The responder will return whether this transmission unit size is supported for MCTP packets that it transmits for the given message type.

2134 **12.18 Resolve UUID**

2135 This command is used to get information about an endpoint based on its UUID. This command may be
2136 sent from any endpoint to the bus owner. This command takes a UUID as a parameter in the request and
2137 returns a list of EIDs and physical addresses that matches this UUID.

2138 A bus owner that supports this command shall keep in the routing table entries the UUID of each of the
2139 endpoints. The UUID values can be found using a "Get Endpoint UUID" command.

2140 An endpoint knows the physical address of the bus owner by keeping track of which physical address
2141 was used when the endpoint received its EID assignment through the Set Endpoint ID command. The
2142 endpoint can send this command to the bus owner using the null destination EID value. This eliminates
2143 the need for the endpoint to also keep track of the EID of the bus owner. The request and response
2144 parameters are specified in Table 33.

2145 **Table 33 – Resolve UUID message**

	Byte	Description
Request data	1:16	Requested UUID
	17	Entry Handle (0x00 to access first entries in table)

	Byte	Description
Response data	1	Completion Code
	2	Next Entry Handle (Use this value to request the next set of entries, if any.) If the EID table data exceeds what can be carried in a single MCTP control response. 0xFF = No more entries
	3	Number of EID entries being returned in this response.
	4:N	One or more routing table entries, formatted per Table 34. This field will be absent if the number of EID entries is 0x00.

2146

Table 34 – Resolve UUID message entry format

Byte	Description
0	EID
1	Physical Transport Binding Type Identifier, according to MCTP ID specification (DSP0239).
2	Physical Media Type Identifier, according to MCTP ID specification (DSP0239). This value is used to indicate what format the following physical address data is given in.
3	Physical Address Size.
4:N	Physical Address.

2147 12.19 Query rate limit

2148 This command can be used to query an EID for its transmit rate limiting capabilities and its receive data
2149 rate requirements.

2150 This command can be used by a message originator to determine the data rate that this EID accepts. The
2151 command can also be used to query the present settings for the EID's transmit data rate capabilities and
2152 present setting.

2153 The request and response parameters are specified in Table 35.

2154

Table 35 – Query rate limit message

	Byte	Description
Request data	-	-
Response data	1	Completion Code
	2:5	Receive information: receive buffer size in bytes.
	6:9	Receive Information: maximum receive data rate limit, in baseline transmission unit packets/sec. A value of 0x0 indicates the receiver is not requesting limiting of the traffic. Note: Unless otherwise specified, it should be assumed that the limit has been defined for communication between two EIDs with the receiver in its most typical modes of operation. The value is not a guarantee. Factors such as transient loading, and a typical device states may mean the receiver will be temporarily unable to receive at the rate given in this response.
	10:13	Transmit Rate limiter capabilities: Maximum supported rate limit, in baseline transmission unit packets/sec. A value of 0x0 means the device cannot throttle its traffic.

	Byte	Description
	14:17	Transmit Rate limiter capabilities: Minimum supported rate limit, in baseline transmission unit packets/sec. A value of 0x0 means the device cannot throttle its traffic.
	18:20	Transmit Rate limiter capabilities: Maximum supported burst size.
	21:23	Present Transmit Rate Limit Burst Setting: The maximal burst size allowed to be sent from this EID at one time.
	24:27	Present Setting: EID Maximal Transmit data rate limit, in baseline transmission unit packets/sec. A value of 0x0 means the rate limiter is not active (When Rate Limiting is inactive, the EID will be transmitting at the maximum rate for its present state).
	28	Transmit Rate limiter capabilities: [7:2] – Reserved [1] – Transmit Rate limiting operation capability 0b – Transmit Rate limiting on this EID is applied to requested and non-requested messages together 1b – Transmit Rate limiting on this EID is applied only to non-requested messages [0] - Rate limiting Support on EID 0b – Transmit Rate limiting is not supported 1b – Transmit Rate limiting is supported

2155 **12.20 Request TX rate limit**

2156 This command can be used to configure an EID for its maximal transmit rate limitations settings.

2157 This command shall be used by a data-receiving device to request to configure a transmitting EID for the
 2158 maximal allowed data rate from the transmitting endpoint to that data-receiving EID.

2159 The request and response parameters are specified in Table 36.

2160 **Table 36 – Request TX rate limit message**

	Byte	Description
Request data	1:3	EID transmit maximal burst size in in MCTP packets. This value defines the maximum number of back-to-back consecutive packets that are allowed to be sent from this endpoint, which the receiving EID supports. The term ‘back-to-back’ means the packets are transmitted with the minimum delay between them. This value shall be set to at least 1 packet to enable rate-limiting. A value of 0 in this field shall be used only to disable rate-limiting.
	4:7	EID Maximal Transmit data rate limit, in baseline transmission unit packets/sec.
Response data	1	Completion Code An ERROR_INVALID_DATA shall be returned if the rate limit requested is not supported.
	2:5	EID transmit burst size in MCTP packets. This value defines the presently used maximum total burst size allowed to be sent from this endpoint at one time.
	6:9	EID transmit data rate limit, as presently used, in baseline transmission unit packets/sec.

2161 The response values for EID transmit burst size in MCTP packets, and EID transmit data rate limit, may
 2162 differ from the requested values. This can happen when multiple requests from multiple source EIDs
 2163 received with different request values sharing the same rate limiter. See description in 10.1.5.

2164 The response to this command is sent when the new rate is in effect when a change is performed or
 2165 immediately when no change is done. Following sending a response to Request TX rate limit command
 2166 for the first time from any EID, it is recommended that the endpoint receiving this command will send Get
 2167 Endpoint UUID command to the EID which sent the Request TX rate limit command. This allows any
 2168 device to identify when an endpoint is enumerated with a different EID, in order to properly calculate its
 2169 rate-limiting settings.

2170 **12.21 Update rate limit**

2171 This command is sent from a transmitter EID to a receiver EID, to update a receiver on any change in the
 2172 transmitter’s rate settings, which did not originate from a request from the receiver. This command is sent
 2173 to any connected receive EID which is not the EID which originated the rate change.

2174 The command shall be used only after a change of the EID transmit burst size and/or EID transmit data
 2175 rate limit.

2176 The request and response parameters are specified in Table 38.

2177 **Table 37 – Update rate limit message**

	Byte	Description
Request data	1:3	EID transmit burst size in MCTP packets. This value defines the presently used maximum total burst size allowed to be sent from this endpoint at one time.
	4:7	EID transmit data rate limit, as presently used, in baseline transmission unit packets/sec.
Response data	1	Completion Code

2178 If an error occurred on the transmitter which caused the rate limiting to be set to an unsupported rate, the
 2179 receiver EID shall issue a new Request TX rate limit command to the transmitter EID.

2180 **12.22 Query supported interfaces**

2181 This command can be used to query an endpoint for its MCTP interfaces capabilities.

2182 This command can be used by an MCTP device A to query the different interfaces which are available on
 2183 MCTP device B for communicating MCTP messages between device A and B.

2184 The request and response parameters are specified in Table 38.

2185 **Table 38 – Query supported interfaces**

	Byte	Description
Request data	-	-
Response data	1	Completion Code
	2	Supported Interfaces Count (shall be ≥ 1)
	3	First interface Type (see MCTP ID)
	4	First interface EID

	Byte	Description
	...	
	...	
	N-1	Last interface Type (see MCTP ID)
	N	Last interface EID

2186 **12.23 Transport Specific**

2187 Transport Specific commands are a range of commands that are available for use by transport binding
 2188 specifications in order to perform additional MCTP Control functions that are defined by a particular
 2189 transport binding. Transport specific commands shall only be addressed to endpoints on the same
 2190 medium. A bridge is allowed to block transport specific commands from being bridged to different media.

2191 The request and response parameters are specified in Table 39.

2192 **Table 39 – Transport Specific message**

	Byte	Description
Request data	1	MCTP Physical Transport Binding Identifier The ID of the Physical Transport specification that defines the transport specific message. This ID is defined in the MCTP ID companion document to this specification.
	2	MCTP Physical Media Identifier The ID of the physical medium that the message is targeted for. This ID is defined in the MCTP ID companion document to this specification.
	3:N	Transport specific command data. Defined by the transport binding specification identified by the MCTP Physical Transport Binding Identifier given in byte 1. If the Physical Transport Binding Identifier = Vendor Defined: The first four bytes of data shall be the IANA Enterprise ID for the Vendor. MSB first. See 12.8.1 for the information on the IANA Enterprise ID as used in this specification.
Response data	1	Completion Code

2193 **13 Vendor Defined – PCI and Vendor Defined – IANA messages**

2194 The Vendor Defined – PCI and Vendor Defined – IANA message types provide a mechanism for
 2195 providing an MCTP message namespace for vendor-specific messages over MCTP.

2196 The PCI and IANA designations refer to the mechanism that is used to identify the vendor or organization
 2197 this is specifying the message’s functionality and any parametric data or other fields provided in the
 2198 message body.

2199 Note that this specification only defines the initial bytes in the message body of these messages, and sets
 2200 the requirement that these messages shall follow the requirements set by the MCTP base protocol and
 2201 any additional requirements necessary to meet the transport of these messages over a particular
 2202 medium, such as path transmission unit limitations.

2203 Otherwise, any other field definitions and higher-level message behavior such as retries, error/completion
 2204 codes, and so on, is message type-specific and thus is vendor-specific.

2205

2206 **13.1 Vendor Defined – PCI message format**

2207 For these messages, the MCTP message type is set to the value for "Vendor Defined – PCI" as defined in
 2208 Table 3. The request and response parameters are specified in Table 40.

2209 **Table 40 – Vendor Defined – PCI message format**

	Byte	Description
Request data	1:2	PCI/PCIe Vendor ID. Refer to PCIe . MSB first. This value is formatted per the Vendor Data Field for the PCI Express vendor ID format. See 12.8.1". NOTE: Because the vendor ID format is implied by the command, the Vendor ID Format bytes are not part of this field.
	(3:N)	Vendor-Defined Message Body. 0 to N bytes.
Response data	1:2	PCI/PCIe Vendor ID. Refer to PCIe . MSB first.
	(3:M)	Vendor-Defined Message Body. 0 to M bytes.

2210 **13.2 Vendor Defined – IANA message format**

2211 For these messages, the MCTP message type is set to the value for "Vendor Defined – IANA" as defined
 2212 in Table 3. The request and response parameters are specified in Table 41.

2213 **Table 41 – Vendor Defined – IANA message format**

	Byte	Description
Request data	1:4	IANA Enterprise ID for Vendor. MSB first. This value is formatted per the Vendor Data Field for the IANA enterprise vendor ID format. See 12.8.1. NOTE: Because the vendor ID format is implied by the command, the Vendor ID Format bytes are not part of this field.
	(5:N)	Vendor-Defined Message Body. 0 to N bytes.
Response data	1:4	IANA Enterprise ID for the Vendor. MSB first.
	(5:M)	Vendor-Defined Message Body. 0 to M bytes.

2214

ANNEX A (informative)

Notation

2215
2216
2217
2218

2219 A.1 Notations

2220 Examples of notations used in this document are as follows:

- 2221 • 2:N In field descriptions, this will typically be used to represent a range of byte offsets
2222 starting from byte two and continuing to and including byte N. The lowest offset is on
2223 the left, and the highest is on the right.
- 2224 • (6) Parentheses around a single number can be used in message field descriptions to
2225 indicate a byte field that may be present or absent.
- 2226 • (3:6) Parentheses around a field consisting of a range of bytes indicates the entire range
2227 may be present or absent. The lowest offset is on the left, and the highest is on the
2228 right.
- 2229 • PCle Underlined, blue text is typically used to indicate a reference to a document or
2230 specification called out in 2, "Normative References", or to items hyperlinked within
2231 the document.
- 2232 • rsvd Abbreviation for Reserved. Case insensitive.
- 2233 • [4] Square brackets around a number are typically used to indicate a bit offset. Bit offsets
2234 are given as zero-based values (that is, the least significant bit [LSb] offset = 0).
- 2235 • [7:5] A range of bit offsets. The most-significant is on the left, and the least-significant is on
2236 the right.
- 2237 • 1b The lower case "b" following a number consisting of 0s and 1s is used to indicate the
2238 number is being given in binary format.
- 2239 • 0x12A A leading "0x" is used to indicate a number given in hexadecimal format.

2240

ANNEX B (informative)

Change log

Version	Date	Description
1.0.0	2009-05-21	
1.1.0	2010-02-19	Updated the glossary and the overview section, including additions for MCTP host interfaces and descriptions of MCTP networks. Added support for MCTP network IDs and the Get Network ID command. Addressed Mantis issue: 0000417. Added text to Clause 1 (Scope) referencing DSP0238, DSP0239 per WG ballot comments.
1.2.0	2013-01-10	Added <i>Resolve UUID</i> command. Clarified use of Control Protocol Version and versioning for OEM commands, <i>Prepare for Endpoint Discovery</i> command, and the <i>Allocate Endpoint IDs</i> command. Clarified requirements on MCTP Control message flags and TO bit use. Changed command requirements to allow an Endpoint to optionally accept or generate Routing Information Update commands. Corrected typographic and formatting errors.
1.2.1	2014-10-09	Corrected error requiring the TD bit to be set to 0b for MCTP VDMs. TD bit should follow PCIe specifications. Corrected misuse of reserved EIDs in figures. Changed document organization to place bridging clauses in a new first level clause "MCTP Bridging". Added clarifications and clause on "Endpoint ID Retention". Added more cross references and clarifications to better identify requirements associated with the <i>Get Endpoint UUID</i> command.
1.3.0	2016-11-24	Added .Rate Limiting. Fixed formatting errors and typos. Added Query Supported Interfaces command

2246

2247

2248