



1  
2  
3  
4  
5

**Document Number: DSP0236**

**Date: 2013-01-24**

**Version: 1.2.0**

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

9 **Document Type: Specification**

10 **Document Status: DMTF Standard**

11 **Document Language: en-US**

12 Copyright notice

13 Copyright © 2013 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29 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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.....	19
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	8.19 MCTP Bridging.....	40
74	8.20 Bridge and Routing Table Examples .....	47
75	8.21 Endpoint ID Resolution .....	51
76	8.22 Bridge and Bus Owner Implementation Recommendations.....	53
77	8.23 Path and Transmission Unit Discovery.....	54
78	8.24 Path Transmission Unit Requirements for Bridges.....	57
79	9 MCTP Control Protocol .....	57
80	9.1 Terminology .....	57
81	9.2 MCTP Control Message Format.....	58
82	9.3 MCTP Control Message Fields.....	58
83	9.4 MCTP Control Message Transmission Unit Size .....	59
84	9.5 Tag Owner (TO), Request (Rq), and Datagram (D) Bit Usage.....	59
85	9.6 Concurrent Command Processing.....	60
86	10 MCTP Control Messages .....	61
87	10.1 MCTP Control Message Command Codes .....	61
88	10.2 MCTP Control Message Completion Codes.....	63
89	10.3 Set Endpoint ID.....	63

90	10.4	Get Endpoint ID .....	65
91	10.5	Get Endpoint UUID Command .....	66
92	10.6	Get MCTP Version Support .....	67
93	10.7	Get Message Type Support .....	71
94	10.8	Get Vendor Defined Message Support .....	71
95	10.9	Resolve Endpoint ID .....	73
96	10.10	Allocate Endpoint IDs .....	74
97	10.11	Routing Information Update .....	76
98	10.12	Get Routing Table Entries .....	78
99	10.13	Prepare for Endpoint Discovery .....	79
100	10.14	Endpoint Discovery .....	80
101	10.15	Discovery Notify .....	80
102	10.16	Get Network ID Command .....	81
103	10.17	Query Hop .....	82
104	10.18	Resolve UUID .....	83
105	10.19	Transport Specific .....	84
106	11	Vendor Defined – PCI and Vendor Defined – IANA Messages .....	84
107	11.1	Vendor Defined – PCI Message Format .....	85
108	11.2	Vendor Defined – IANA Message Format .....	85
109		ANNEX A (informative) Notation .....	86
110		ANNEX B (informative) Change Log .....	87
111			

## 112 Figures

113	Figure 1	– Management Component Relationships .....	21
114	Figure 2	– MCTP Networks .....	22
115	Figure 3	– MCTP Topology .....	24
116	Figure 4	– Generic Message Fields .....	25
117	Figure 5	– Topmost Bus Owners .....	35
118	Figure 6	– Split Bridge .....	36
119	Figure 7	– Acceptable Failover/Redundant Communication Topologies .....	40
120	Figure 8	– Routing/Bridging Restrictions .....	41
121	Figure 9	– EID Options for MCTP Bridges .....	42
122	Figure 10	– Basic Routing Table Entry Fields .....	45
123	Figure 11	– Routing Table Population .....	46
124	Figure 12	– Example 1 Routing Topology .....	48
125	Figure 13	– Example 2 Routing Topology .....	49
126	Figure 14	– Example 3 Routing Topology .....	50
127	Figure 15	– Endpoint ID Resolution .....	52
128	Figure 16	– Resolving Multiple Paths .....	53
129	Figure 17	– Example Path Routing Topology .....	55
130	Figure 18	– Path Transmission Unit Discovery Flowchart .....	56
131	Figure 19	– MCTP Control Message Format .....	58
132	Figure 20	– Structure of Vendor ID Field for Get Vendor Defined Capabilities Message .....	72
133	Figure 21	– EID Pools from Multiple Bus Owners .....	75

134 **Tables**

135	Table 1 – MCTP Base Protocol Common Fields .....	25
136	Table 2 – Special Endpoint IDs.....	27
137	Table 3 – MCTP Message Types Used in this Specification .....	32
138	Table 4 – Example 1 Routing Table for D2.....	48
139	Table 5 – Example 2 Routing Table for D1.....	49
140	Table 6 – Example 3 Routing Table for D2.....	50
141	Table 7 – Additional Information Tracked by Bridges .....	51
142	Table 8 – MCTP Control Protocol Terminology .....	57
143	Table 9 – MCTP Control Message Types.....	57
144	Table 10 – MCTP Control Message Fields .....	58
145	Table 11 – Tag Owner (TO), Request (Rq) and Datagram (D) Bit Usage.....	59
146	Table 12 – MCTP Control Command Numbers.....	61
147	Table 13 – MCTP Control Message Completion Codes.....	63
148	Table 14 – Set Endpoint ID Message .....	64
149	Table 15 – Get Endpoint ID Message.....	65
150	Table 16 – Get Endpoint UUID Message Format .....	67
151	Table 17 – Example UUID Format.....	67
152	Table 18 – Get MCTP Version Support Message .....	68
153	Table 19 – Get Message Type Support Message .....	71
154	Table 20 – Get Vendor Defined Message Support Message .....	72
155	Table 21 – Vendor ID Formats.....	73
156	Table 22 – Resolve Endpoint ID Message.....	73
157	Table 23 – Allocate Endpoint IDs Message .....	75
158	Table 24 – Routing Information Update Message .....	77
159	Table 25 – Routing Information Update Entry Format .....	77
160	Table 26 – Get Routing Table Entries Message .....	78
161	Table 27 – Routing Table Entry Format.....	78
162	Table 28 – Prepare for Endpoint Discovery Message .....	80
163	Table 29 – Endpoint Discovery Message .....	80
164	Table 30 – Discovery Notify Message .....	81
165	Table 31 – Get Network ID Message Format .....	81
166	Table 32 – Query Hop Message .....	82
167	Table 12 – Resolve UUID Message.....	83
168	Table 13 – Resolve UUID Message Entry Format.....	83
169	Table 33 – Transport Specific Message .....	84
170	Table 34 – Vendor Defined – PCI Message Format.....	85
171	Table 35 – Vendor Defined – IANA Message Format .....	85
172		

173

## Foreword

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

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

178

## Introduction

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

- 181 • Management controllers and other management controllers
- 182 • Management controllers and management devices

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

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

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

194 It is intended that different types of devices in a management system may need to implement different  
195 portions of the complete capabilities defined by this protocol. Where relevant, this is called out in the  
196 individual requirements  
197





# 199 Management Component Transport Protocol (MCTP) Base 200 Specification

## 201 1 Scope

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

205 This document is intended to meet the following objectives:

- 206 • Describe the MCTP Base transport protocol
- 207 • Describe the MCTP control message protocol

208 The MCTP specifies a transport protocol format. This protocol is independent of the underlying physical  
209 bus properties, as well as the "data-link" layer messaging used on the bus. The physical and data-link  
210 layer methods for MCTP communication across a given medium are defined by companion "transport  
211 binding" specifications, such as [DSP0238](#), MCTP over PCIe® Vendor Defined Messaging, and [DSP0237](#),  
212 MCTP over SMBus/ I<sup>2</sup>C. This approach enables future transport bindings to be defined to support  
213 additional buses such as USB, RMII, and others, without affecting the base MCTP specification.

## 214 2 Normative References

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

### 218 2.1 Approved References

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

220 [http://www.dmtf.org/standards/published\\_documents/DSP4004\\_2.7.0.pdf](http://www.dmtf.org/standards/published_documents/DSP4004_2.7.0.pdf)

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

222 [http://www.dmtf.org/standards/published\\_documents/DSP2016.pdf](http://www.dmtf.org/standards/published_documents/DSP2016.pdf)

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

224 [http://www.dmtf.org/standards/published\\_documents/DSP0239\\_1.2.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0239_1.2.0.pdf)

### 225 2.2 Other References

226 DMTF DSP0237, Management Component Transport Protocol SMBus/I2C Transport Binding  
227 Specification

228 [http://www.dmtf.org/standards/published\\_documents/DSP0237\\_1.0.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0237_1.0.0.pdf)

229 DMTF DSP0238, Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding  
230 Specification

231 [http://www.dmtf.org/standards/published\\_documents/DSP0238\\_1.0.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0238_1.0.0.pdf)

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

234 <http://www.acpi.info/DOWNLOADS/ACPIspec50.pdf>

- 235 IETF, RFC20, *ASCII format for Network Interchange*, October 16, 1969  
236 <http://tools.ietf.org/html/rfc20>
- 237 IETF, RFC4122, *A Universally Unique Identifier (UUID) URN Namespace*, July 2005  
238 <http://datatracker.ietf.org/doc/rfc4122/>
- 239 IETF, RFC2119, *Key Words for use in RFCs to Indicate Requirement Levels*, March 1997  
240 <http://datatracker.ietf.org/doc/rfc2119/>
- 241 Intel, Hewlett-Packard, NEC, and Dell, *Intelligent Platform Management Interface Specification: Second Generation v2.0*, IPMI, 2004  
242  
243 <http://www.intel.com/design/servers/ipmi>
- 244 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*  
245 <http://isotc.iso.org/livelink/livelink?func=ll&objId=4230456&objAction=browse&sort=subtype>
- 246 PCI-SIG, PCI Express™ Specifications  
247 <http://www.pcisig.com/specifications/pciexpress/>
- 248 NXP Semiconductors, *UM10204 I2C-bus specification and user manual*, Rev. 5, October 9, 2012  
249 [http://www.nxp.com/documents/user\\_manual/UM10204.pdf](http://www.nxp.com/documents/user_manual/UM10204.pdf)
- 250 SMBus, *System Management Bus (SMBus) Specification v2.0*, SMBus, 2000  
251 <http://www.smbus.org/specs/smbus20.pdf>

## 252 3 Terms and Definitions

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

### 254 3.1 Requirement Term Definitions

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

#### 257 3.1.1

##### 258 **can**

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

#### 260 3.1.2

##### 261 **cannot**

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

#### 263 3.1.3

##### 264 **conditional**

265 indicates requirements to be followed strictly to conform to the document when the specified conditions  
266 are met

#### 267 3.1.4

##### 268 **deprecated**

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

- 270 **3.1.5**  
271 **mandatory**  
272 indicates requirements to be followed strictly to conform to the document and from which no deviation is  
273 permitted
- 274 **3.1.6**  
275 **may**  
276 indicates a course of action permissible within the limits of the document
- 277 NOTE: An implementation that does *not* include a particular option must be prepared to interoperate with another  
278 implementation that *does* include the option, although perhaps with reduced functionality. An implementation that  
279 *does* include a particular option must be prepared to interoperate with another implementation that does *not* include  
280 the option (except for the feature that the option provides).
- 281 **3.1.7**  
282 **may not**  
283 indicates flexibility of choice with no implied preference
- 284 **3.1.8**  
285 **must**  
286 indicates that the item is an absolute requirement of the specification
- 287 **3.1.9**  
288 **must not**  
289 indicates that the definition is an absolute prohibition of the specification
- 290 **3.1.10**  
291 **need not**  
292 indicates a course of action permissible within the limits of the document
- 293 **3.1.11**  
294 **not recommended**  
295 indicates that valid reasons may exist in particular circumstances when the particular behavior is  
296 acceptable or even useful, but the full implications should be understood and carefully weighed before  
297 implementing any behavior described with this label
- 298 **3.1.12**  
299 **obsolete**  
300 indicates that an item was defined in prior specifications but has been removed from this specification
- 301 **3.1.13**  
302 **optional**  
303 indicates a course of action permissible within the limits of the document
- 304 **3.1.14**  
305 **recommended**  
306 indicates that valid reasons may exist in particular circumstances to ignore a particular item, but the full  
307 implications should be understood and carefully weighed before choosing a different course

- 308 **3.1.15**  
309 **required**  
310 indicates that the item is an absolute requirement of the specification
- 311 **3.1.16**  
312 **shall**  
313 indicates requirements to be followed strictly to conform to the document and from which no deviation is  
314 permitted
- 315 **3.1.17**  
316 **shall not**  
317 indicates requirements to be followed strictly to conform to the document and from which no deviation is  
318 permitted
- 319 **3.1.18**  
320 **should**  
321 indicates that among several possibilities, one is recommended as particularly suitable, without  
322 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 323 **3.1.19**  
324 **should not**  
325 indicates that a certain possibility or course of action is deprecated but not prohibited  
326

## 327 **3.2 MCTP Term Definitions**

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

- 329 **3.2.1**  
330 **Address Resolution Protocol**  
331 **ARP**  
332 refers to the procedure used to dynamically determine the addresses of devices on a shared  
333 communication medium
- 334 **3.2.2**  
335 **baseline transmission unit**  
336 the required common denominator size of a transmission unit for packet payloads that are carried in an  
337 MCTP packet. Baseline Transmission Unit-sized packets are guaranteed to be routable within an MCTP  
338 network.
- 339 **3.2.3**  
340 **baseboard management controller**  
341 **BMC**  
342 a term coined by the IPMI specifications for the main management controller in an IPMI-based platform  
343 management subsystem. Also sometimes used as a generic name for a motherboard resident  
344 management controller that provides motherboard-specific hardware monitoring and control functions for  
345 the platform management subsystem.

346 **3.2.4**347 **binary-coded decimal**348 **BCD**

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

357 **3.2.5**358 **bridge**

359 generically, the circuitry and logic that connects one computer bus or interconnect to another, allowing an  
360 agent on one to access the other. Within this document, the term *bridge* shall refer to MCTP bridge,  
361 unless otherwise indicated.

362 **3.2.6**363 **bus**

364 a physical addressing domain shared between one or more platform components that share a common  
365 physical layer address space

366 **3.2.7**367 **bus owner**

368 the party responsible for managing address assignments (can be logical or physical addresses) on a bus  
369 (for example, in MCTP, the bus owner is the party responsible for managing EID assignments for a given  
370 bus). A bus owner may also have additional media-specific responsibilities, such as assignment of  
371 physical addresses.

372 **3.2.8**373 **byte**

374 an 8-bit quantity. Also referred to as an *octet*.

375 NOTE: PMCI specifications shall use the term *byte*, not *octet*.

376 **3.2.9**377 **endpoint**

378 see [MCTP endpoint](#)

379 **3.2.10**380 **endpoint ID**381 **EID**

382 see [MCTP endpoint ID](#)

383 **3.2.11**384 **Globally Unique Identifier**385 **GUID**

386 see [UUID](#)

**387 3.2.12****388 host interface**

389 a hardware interface and associated protocols that is used by software running locally on the host  
390 processors to access the hardware of a management subsystem within a managed system.

**391 3.2.13****392 Inter-Integrated Circuit****393 I<sup>2</sup>C**

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

**396 3.2.14****397 intelligent management device****398 IMD**

399 a management device that is typically implemented using a microcontroller and accessed through a  
400 messaging protocol. Management parameter access provided by an IMD is typically accomplished using  
401 an abstracted interface and data model rather than through direct "register level" accesses.

**402 3.2.15****403 Intelligent Platform Management Bus****404 IPMB**

405 name for the architecture, protocol, and implementation of an I<sup>2</sup>C bus that provides a communications  
406 path between "management controllers" in IPMI -based systems

**407 3.2.16****408 Intelligent Platform Management Interface****409 IPMI**

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

**412 3.2.17****413 managed entity**

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

**417 3.2.18****418 Management Component Transport Protocol****419 MCTP**

420 The protocol defined in this specification.

**421 3.2.19****422 management controller**

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

429 or processor that serves as a management controller can also incorporate the functions of a management  
430 device.

### 431 **3.2.20**

#### 432 **management device**

433 any physical device that provides protocol terminus for accessing one or more management parameters.  
434 A management device responds to management requests, but does not initiate or aggregate  
435 management operations except in conjunction with a management controller (that is, it is a *satellite*  
436 device that is subsidiary to one or more management controllers). An example of a simple management  
437 device would be a temperature sensor chip. A management controller that has I/O pins or built-in analog-  
438 to-digital converters that monitor state and voltages for a managed entity would also be a management  
439 device.

### 440 **3.2.21**

#### 441 **management parameter**

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

### 445 **3.2.22**

#### 446 **MCTP bridge**

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

### 450 **3.2.23**

#### 451 **MCTP bus owner**

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

### 456 **3.2.24**

#### 457 **MCTP control command**

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

### 461 **3.2.25**

#### 462 **MCTP endpoint**

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

### 467 **3.2.26**

#### 468 **MCTP endpoint ID**

469 the logical address used to route MCTP messages to a specific MCTP endpoint. A numeric handle  
470 (logical address) that uniquely identifies a particular MCTP endpoint within a system for MCTP  
471 communication and message routing purposes. Endpoint IDs are unique among MCTP endpoints that  
472 comprise an MCTP communication network within a system. MCTP EIDs are only unique within a

473 particular MCTP network. That is, they can be duplicated or overlap from one MCTP network to the next.  
474 Also referred to in this document as "endpoint ID" and abbreviated "EID".

### 475 **3.2.27**

#### 476 **MCTP host interface**

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

### 478 **3.2.28**

#### 479 **MCTP management controller**

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

### 482 **3.2.29**

#### 483 **MCTP management device**

484 a management device that is an MCTP endpoint. Unless otherwise indicated, the term "management  
485 device" refers to an "MCTP management device" in this document.

### 486 **3.2.30**

#### 487 **MCTP message**

488 a unit of communication based on the message type that is relayed through the MCTP Network using one  
489 or more MCTP packets

### 490 **3.2.31**

#### 491 **MCTP network**

492 a collection of MCTP endpoints that communicate using MCTP and share a common MCTP endpoint ID  
493 space

### 494 **3.2.32**

#### 495 **MCTP network ID**

496 a unique identifier to distinguish each independent MCTP network within a platform

### 497 **3.2.33**

#### 498 **MCTP packet**

499 the unit of data transfer used for MCTP communication on a given physical medium

### 500 **3.2.34**

#### 501 **MCTP packet payload**

502 refers to the portion of the message body of an MCTP message that is carried in a single MCTP packet

### 503 **3.2.35**

#### 504 **message**

505 see [MCTP message](#)

### 506 **3.2.36**

#### 507 **message assembly**

508 the process of receiving and linking together two or more MCTP packets that belong to a given MCTP  
509 message to allow the entire message header and message data (payload) to be extracted



- 510 **3.2.37**  
511 **message body**  
512 the portion of an MCTP message that carries the message type field and any message type-specific data  
513 associated with the message. An MCTP message spans multiple MCTP packets when the message body  
514 needs is larger than what can fit in a single MCTP packet. Thus, the message body portion of an MCTP  
515 message can span multiple MCTP packets.
- 516 **3.2.38**  
517 **message disassembly**  
518 the process of taking an MCTP message where the message's header and data (payload) cannot be  
519 carried in a single MCTP packet and generating the sequence of two or more packets required to deliver  
520 that message content within the MCTP network
- 521 **3.2.39**  
522 **message originator**  
523 the original transmitter (source) of a message targeted to a particular message terminus
- 524 **3.2.40**  
525 **message terminus**  
526 the name for a triplet of fields called the MCTP Source Endpoint ID, Tag Owner bit value, and Message  
527 Tag value. Together, these fields identify the packets for an MCTP message within an MCTP network for  
528 the purpose of message assembly. The message terminus itself can be thought of as identifying a set of  
529 resources within the recipient endpoint that is handling the assembly of a particular message.
- 530 **3.2.41**  
531 **most significant byte**  
532 **MSB**  
533 refers to the highest order byte in a number consisting of multiple bytes
- 534 **3.2.42**  
535 **nibble**  
536 the computer term for a four-bit aggregation, or half of a byte
- 537 **3.2.43**  
538 **packet**  
539 see [MCTP packet](#)
- 540 **3.2.44**  
541 **packet payload**  
542 see [MCTP packet payload](#)
- 543 **3.2.45**  
544 **pass-through traffic/message/packets**  
545 non-control packets passed between the external network and the management controller through the  
546 network controller
- 547 **3.2.46**  
548 **payload**  
549 refers to the information bearing fields of a message. This is separate from those fields and elements that  
550 are used to transport the message from one point to another, such as address fields, framing bits,  
551 checksums, and so on. In some instances, a given field may be both a payload field and a transport field.

- 552 **3.2.47**  
553 **physical transport binding**  
554 refers to specifications that define how the MCTP base protocol and MCTP control commands are  
555 implemented on a particular physical transport type and medium, such as SMBus/I<sup>2</sup>C, PCI Express™  
556 Vendor Defined Messaging, and so on.
- 557 **3.2.48**  
558 **Platform Management Component Intercommunications**  
559 **PMCI**  
560 name for a working group under the Distributed Management Task Force's Pre-OS Workgroup that is  
561 chartered to define standardized communication protocols, low level data models, and transport  
562 definitions that support communications with and between management controllers and management  
563 devices that form a platform management subsystem within a managed computer system
- 564 **3.2.49**  
565 **point-to-point**  
566 refers to the case where only two physical communication devices are interconnected through a physical  
567 communication medium. The devices may be in a master/slave relationship, or could be peers.
- 568 **3.2.50**  
569 **Reduced Media Independent Interface**  
570 **RMII**  
571 a reduced signal count MAC to PHY interface, based on the IEEE Media Independent Interface (MII),  
572 which was specified by the RMII Consortium (3Com Corporation; AMD Inc.; Bay Networks, Inc.;  
573 Broadcom Corp.; National Semiconductor Corp.; and Texas Instruments Inc.)
- 574 **3.2.51**  
575 **simple endpoint**  
576 an MCTP endpoint that is not associated with either the functions of an MCTP bus owner or an MCTP  
577 bridge
- 578 **3.2.52**  
579 **Transmission Unit**  
580 refers to the size of the portion of the MCTP packet payload, which is the portion of the message body  
581 carried in an MCTP packet
- 582 **3.2.53**  
583 **transport binding**  
584 see [physical transport binding](#)
- 585 **3.2.54**  
586 **Universally Unique Identifier**  
587 **UUID**  
588 refers to an identifier originally standardized by the Open Software Foundation (OSF) as part of the  
589 Distributed Computing Environment (DCE). UUIDs are created using a set of algorithms that enables  
590 them to be independently generated by different parties without requiring that the parties coordinate to  
591 ensure that generated IDs do not overlap. In this specification, [RFC4122](#) is used as the base specification  
592 describing the format and generation of UUIDs. Also sometimes referred to as a globally unique identifier  
593 (GUID).

## 594 **4 Symbols and Abbreviated Terms**

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

### 596 **4.1**

#### 597 **ACPI**

598 Advanced Configuration and Power Interface

### 599 **4.2**

#### 600 **ARP**

601 Address Resolution Protocol

### 602 **4.3**

#### 603 **BCD**

604 binary-coded decimal

### 605 **4.4**

#### 606 **BMC**

607 baseboard management controller

### 608 **4.5**

#### 609 **CIM**

610 Common Information Model

### 611 **4.6**

#### 612 **EID**

613 endpoint identifier

### 614 **4.7**

#### 615 **FIFO**

616 first-in first-out

### 617 **4.8**

#### 618 **GUID**

619 Globally Unique Identifier

### 620 **4.9**

#### 621 **I<sup>2</sup>C**

622 Inter-Integrated Circuit

### 623 **4.10**

#### 624 **IANA**

625 Internet Assigned Numbers Authority

### 626 **4.11**

#### 627 **IMD**

628 intelligent management device

### 629 **4.12**

#### 630 **IP**

631 Internet Protocol

- 632 **4.13**  
633 **IPMB**  
634 Intelligent platform management bus
- 635 **4.14**  
636 **IPMI**  
637 Intelligent platform management interface
- 638 **4.15**  
639 **ISO/IEC**  
640 International Organization for Standardization/International Engineering Consortium
- 641 **4.16**  
642 **KCS**  
643 Keyboard Controller Style
- 644 **4.17**  
645 **MCTP**  
646 Management Component Transport Protocol
- 647 **4.18**  
648 **MSB**  
649 most significant byte
- 650 **4.19**  
651 **PCIe**  
652 Peripheral Component Interconnect (PCI) Express
- 653 **4.20**  
654 **PMCI**  
655 Platform Management Component Intercommunications
- 656 **4.21**  
657 **RMII**  
658 Reduced Media Independent Interface
- 659 **4.22**  
660 **SMBus**  
661 System Management Bus
- 662 **4.23**  
663 **TCP/IP**  
664 Transmission Control Protocol/Internet Protocol
- 665 **4.24**  
666 **USB**  
667 Universal Serial Bus
- 668 **4.25**  
669 **UUID**  
670 Universally Unique Identifier

671 **4.26**  
 672 **VDM**  
 673 Vendor Defined Message

674 **5 Conventions**

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

676 **5.1 Byte Ordering**

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

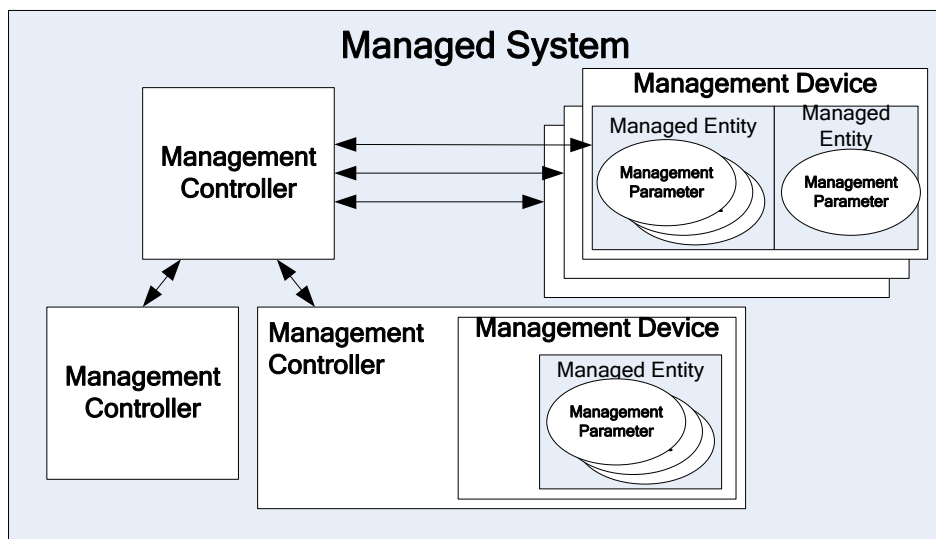
679 **5.2 Reserved Fields**

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

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

684 **6 Management Component Relationships**

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



687

688 **Figure 1 – Management Component Relationships**

689 **7 MCTP Overview**

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

692 MCTP is a transport independent protocol that is used for intercommunication within an MCTP Network.  
 693 An MCTP Network that consists of one or more physical transports that are used to transfer MCTP  
 694 Packets between MCTP Endpoints. MCTP Transport Binding Specifications define how the MCTP  
 695 protocol is implemented across a particular physical transport medium. For example, the DMTF has  
 696 defined transport bindings for MCTP over [SMBus/I<sup>2</sup>C](#) and MCTP over PCIe using PCIe Vendor Defined  
 697 Messages (VDMs), and others.

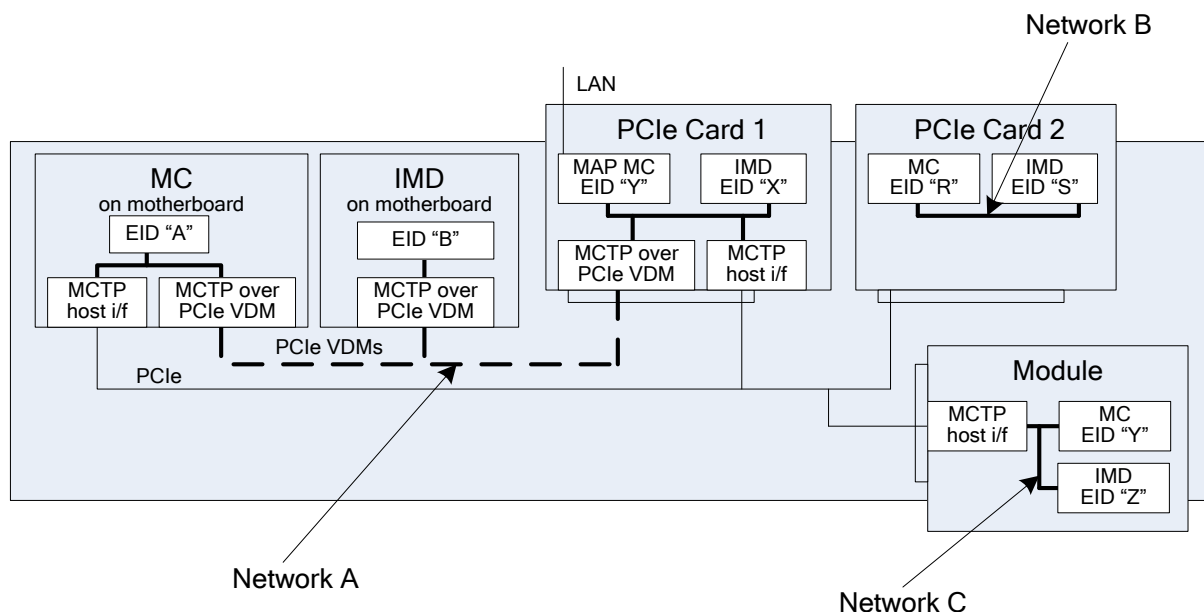
698 An MCTP Endpoint is the terminus for MCTP communication. A physical device that supports MCTP may  
 699 provide one or more MCTP Endpoints. Endpoints are addressed using a logical address called the  
 700 Endpoint ID, or EID. EIDs in MCTP are analogous to IP Addresses in Internet Protocol networking.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

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

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

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

713



714

715

**Figure 2 – MCTP Networks**

716 Figure 2 shows the different ways MCTP Networks can exist in a system. In this example, Network A  
 717 connects a Management Controller (MC) and Intelligent Management Device (IMD) on a motherboard  
 718 with devices on PCIe Card 1 using MCTP over PCIe Vendor Defined Messages. Note that there are two  
 719 host interfaces (host i/f) on standard PCIe that can be used by host software to access this particular

720 network. This network thus requires an MCTP Network ID so that the host software can tell that the two  
721 host interfaces connect to the same MCTP Network.

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

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

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

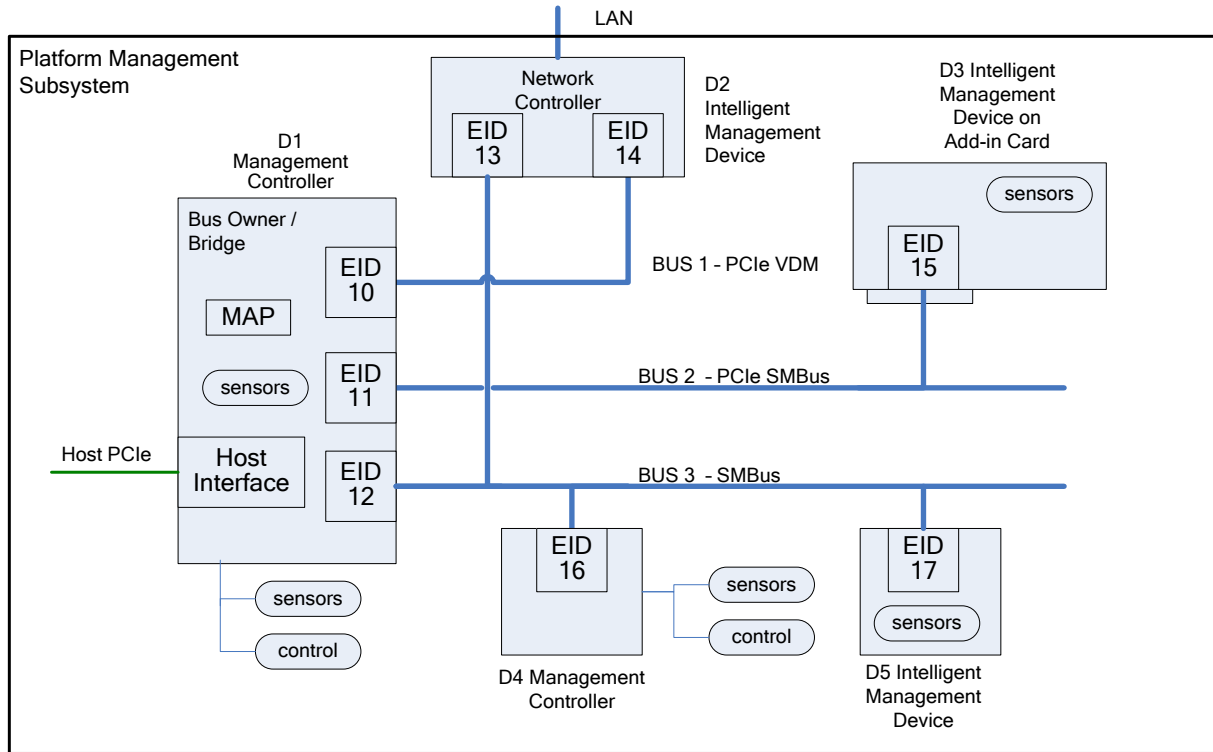
731 MCTP is designed to be able to transfer multiple Message Types in an interleaved manner using the  
732 same protocol. MCTP Message Types are identified using a Message Type number. The use of the message  
733 type number is similar to a well-known port number in Internet Protocol. It identifies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

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

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

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

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

757

758

Figure 3 – MCTP Topology



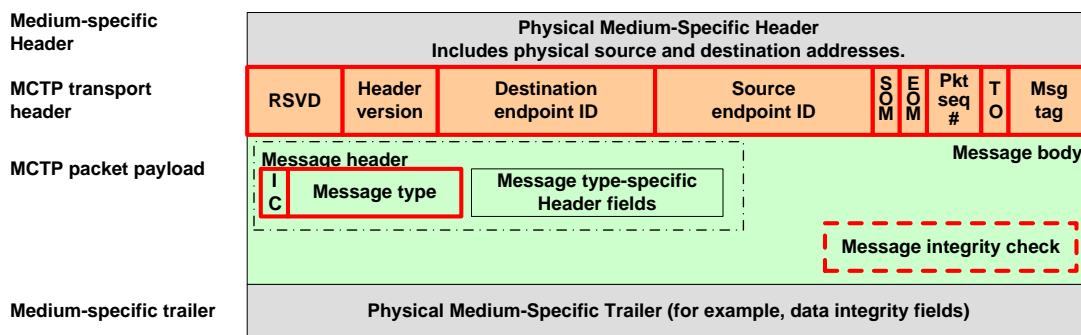
## 759 8 MCTP Base Protocol

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

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

### 766 8.1 MCTP Packet Fields

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



768

769 **Figure 4 – Generic Message Fields**

770 Table 1 defines the base protocol common fields.

771 **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, MCTP over PCIe, 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 must 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. 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 <a href="#">DSP0239</a> 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 must 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 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. This is left to the message type specification.

## 772 8.2 Special Endpoint IDs

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

774 **Table 2 – Special Endpoint IDs**

Value	Description
Destination endpoint ID 0	<b>Null Destination EID.</b> 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	<b>Null Source EID.</b> This value indicates a message is coming from an endpoint that is using physical addressing only. This would typically be used for messages that are delivered from an endpoint that has not been assigned an EID. Because the physical addresses between buses are not guaranteed to be unique, MCTP does not support bridging messages with a null source EID between different buses.
Endpoint IDs 1 through 7	Reserved for future definition.
Endpoint ID 0xFF	<b>Broadcast EID.</b> 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.
------------------	---

### 775 8.3 Packet Payload and Transmission Unit Sizes

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

#### 778 8.3.1 Baseline Transmission Unit

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

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

### 799 8.4 Maximum Message Body Sizes

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

### 802 8.5 Message Assembly

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

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

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

## 8.6 Dropped Packets

810 Individual packets are dropped (silently discarded) by an endpoint under the following conditions. These  
811 packets are discarded before being checked for acceptance or rejection for message assembly.  
812 Therefore, these packets will *not* cause a message assembly to be started or terminated.  
813

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

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

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

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

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

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

- 829 • **Unknown destination EID**

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

- 832 • **Un-routable EID**

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

- 835 • **Bad header version**

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

- 837 • **Unsupported transmission unit**

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

## 8.7 Starting Message Assembly

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

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

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

## 8.8 Terminating Message Assembly/Dropped Messages

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

- 849       •   **Receipt of the "end" packet for the given message**
- 850           Receiving an "end" packet (packet with EOM = 1b) for a message that is in the process of being  
851           assembled on a given message terminus will cause the message assembly to be completed  
852           (provided that the message has not been terminated for any of the reasons listed below). This is  
853           normal termination. The message is considered to be accepted at the MCTP base protocol  
854           level.
- 855       •   **Receipt of a new "start" packet**
- 856           Receiving a new "start" packet (packet with SOM = 1b) for a message to the same message  
857           terminus as a message assembly already in progress will cause the message assembly in  
858           process to be terminated. All data for the message assembly that was in progress is dropped.  
859           The newly received start packet is not dropped, but instead it begins a new message assembly.  
860           This is considered an error condition.
- 861       •   **Timeout waiting for a packet**
- 862           Too much time occurred between packets of a given multiple-packet message. The timeout  
863           interval is specific to a particular medium. All data for the message assembly that was in  
864           progress are dropped. This is considered an error condition.
- 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 control commands for initialization and version support discovery.  
906

## 8.11 MCTP Message Types

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

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

**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, <a href="http://www.iana.org">www.iana.org</a> , 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.

## 8.12 Security

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

## 8.13 Limitations

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

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



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

## 944 8.14 MCTP Discovery and Addressing

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

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

- 948 1) Bus enumeration
- 949 2) Bus address assignment
- 950 3) MCTP capability discovery
- 951 4) Endpoint ID assignment
- 952 5) Distribution and use of routing information

953 This section gives an overview of the methods used for accomplishing each of these steps in various  
954 operational scenarios. Clause 10 gives details on the messages used to implement these operations.

### 955 8.14.1 Bus Enumeration

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

### 959 8.14.2 Bus Address Assignment

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

### 965 8.14.3 MCTP Capability Discovery

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

#### 970 **8.14.4 Endpoint ID Assignment**

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

#### 974 **8.14.5 Distribution and Use of Routing Information**

975 Bridging-capable MCTP endpoints need routing information to identify the next hop to forward a message  
976 to its final destination. See 8.19 for a description of how routing information is conveyed between MCTP  
977 endpoints.

### 978 **8.15 Devices with Multiple Media Interfaces**

979 MCTP fully supports management controllers or management devices that have interfaces on more than  
980 one type of bus. For example, a device could have both a PCI Express (PCIe) and an SMBus interface. In  
981 this scenario, the device will typically have a different EID for each interface. (Bridges can include  
982 instantiations that have an endpoint shared across multiple interfaces; see 8.19.2 for more information.)

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

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

### 990 **8.16 Peer Transactions**

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

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

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

999 It is important to note that in cases where two endpoints are on the same bus, they do not need to go  
1000 through a bridge to communicate with each other. Devices use the Resolve Endpoint ID command to ask  
1001 the bus owner what physical address should be used to route messages to a given EID. Depending on  
1002 the bus implementation, the bus owner can either return the physical address of the bridge that the  
1003 message should be delivered to, or it can return the physical address of the peer on the bus.

### 1004 **8.17 Endpoint ID Assignment and Endpoint ID Pools**

1005 MCTP EIDs are the system-wide unique IDs used by the MCTP infrastructure to address endpoints and  
1006 for routing messages across multiple buses in the system. There is one EID assigned to a given physical  
1007 address. Most intelligent management devices (IMDs) or management controllers will connect to just a  
1008 single bus and have a single EID. A non-bridge device that is connected to multiple different buses will  
1009 have one EID for each bus it is attached to.

1010 Bus owners are MCTP devices that are responsible for issuing EIDs to devices on a bus segment. These  
1011 EIDs come from a pool of EIDs maintained by the bus owner.

1012 With the exception of the topmost bus owner (see 8.17.1), a given bus owner's pool of EIDs is  
 1013 dynamically allocated at run-time by the bus owner of the bus above it in the hierarchy. Hot-plug devices  
 1014 must have their EID pools dynamically allocated.

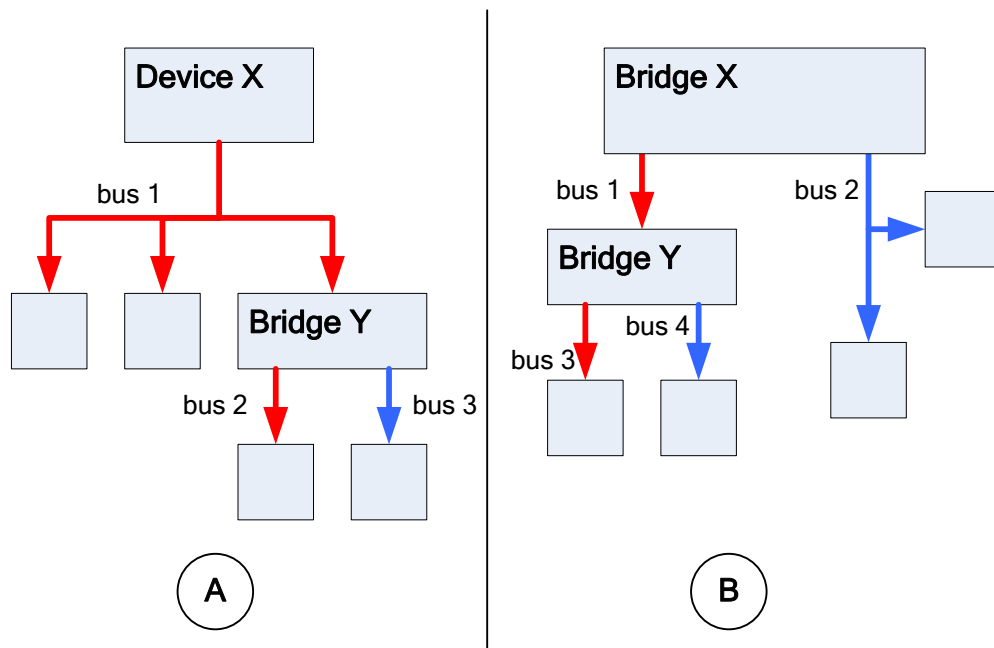
1015 Once EIDs are assigned to MCTP endpoints, it is necessary for MCTP devices involved in a transaction  
 1016 to understand something about the route a given message will traverse. Section 8.19 describes how this  
 1017 routing information is shared among participants along a message's route.

1018 **8.17.1 Topmost Bus Owner**

1019 The topmost bus owner is the ultimate source of the EID pool from which all EIDs are drawn for a given  
 1020 MCTP network.

1021 This is illustrated in Figure 5, in which the arrows are used to identify the role of bus ownership. The  
 1022 arrows point outward from the bus owner for the particular bus and inward to a device that is "owned" on  
 1023 the bus.

1024 In Figure 5, device X in diagram A and bridge X in diagram B are examples of topmost bus owners.  
 1025 Diagram A shows a device that connects to a single bus and is the topmost bus owner for the overall  
 1026 MCTP network. Diagram B shows that a bridge can simultaneously be the topmost bus owner, as well as  
 1027 shows the bus owner for more than one bus. The different colors represent examples of different media.



1028

1029

**Figure 5 – Topmost Bus Owners**

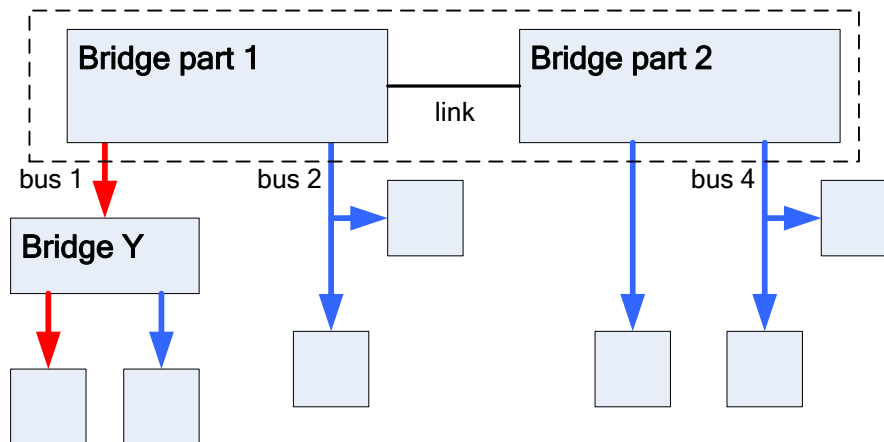


Figure 6 – Split Bridge

1030

1031

1032 An implementation may need to split a bus owner or bridge across two physical devices. Such an  
 1033 implementation must include a mechanism (for example, a link as shown in Figure 6) that enables the two  
 1034 parts to share a common routing table, or have individual copies of the routing table that are kept  
 1035 synchronized. The definition of this mechanism is outside the scope of this specification.

1036

### 1037 8.17.2 Use of Static EIDs and Static EID Pools

1038 In general, the only device that will require a static (pre-configured) EID assignment will be the topmost  
 1039 bus owner. It needs a static EID because there is no other party to assign it an EID through MCTP.  
 1040 Otherwise, all other devices will have their EIDs assigned to them by a bus owner.

1041 The same principle applies if the device functions as an MCTP bridge. If the device is the highest device  
 1042 in the MCTP bus hierarchy, it will require a static pool of EIDs to be assigned to it as part of the system  
 1043 design. Otherwise, the device will be dynamically allocated a pool of EIDs from a higher bus owner.

1044 An MCTP network implementation is allowed to use static EIDs for devices other than the topmost bus  
 1045 owner. Typically, this would only be done for very simple MCTP networks. Other key EID assignment  
 1046 considerations follow:

- 1047 • Endpoints that support the option of being configured for one or more static EIDs must also  
 1048 support being configured to be dynamically assigned EIDs.
- 1049 • No mechanism is defined in the MCTP base specification for a bridge or bus owner to discover  
 1050 and incorporate a static EID into its routing information. Thus, a simple endpoint that is  
 1051 configured with a static EID must also be used with a bus owner that is configured to support  
 1052 the static EIDs for the endpoint.
- 1053 • All bus owners/bridges in the hierarchy, from the topmost bus owner to the endpoint, must have  
 1054 their routing configurable to support static EID routing information.
- 1055 • Although an endpoint that uses a static EID must be used with a bus owner that supports static  
 1056 EIDs, the reverse is not true. A bus owner that uses static EIDs does not need to require that  
 1057 the devices on the buses it owns be configured with static EIDs.
- 1058 • How the configuration of static EIDs occurs is outside the scope of this specification.

- 1059 • No specified mechanism exists to "force" an override of a bridge's or bus owner's routing table  
1060 entries for static EIDs. That is, commands such as Allocate Endpoint IDs and Routing  
1061 Information Update only affect entries that are associated with dynamic EIDs.
- 1062 • MCTP does not define a mechanism for keeping routing tables updated if static EIDs are used  
1063 with dynamic physical addresses. That is, static EIDs are not supported for use with dynamic  
1064 physical addresses.
- 1065 • Bridges can have a mix of both static and dynamic EID pools. That is, the routing table can  
1066 have both static and dynamic entries and can allocate from static and dynamic EID pools. Only  
1067 the dynamic EID pool is given to the bridge by the bus owner using the Allocate Endpoint IDs  
1068 command. There is no specification for how a static EID pool gets configured or how a bridge  
1069 decides whether to give an endpoint an EID from a static or dynamically obtained EID pool.  
1070 There is also no MCTP-defined mechanism to read the static EID pool setting from the bridge.
- 1071 • MCTP bridges and bus owners (except the topmost bus owner) are not required to include  
1072 support for static EIDs.
- 1073 • MCTP does not define a mechanism for allocating EID pools that take static EID assignments  
1074 into account. That is, a bridge cannot request a particular set of EIDs to be allocated to it.
- 1075 • MCTP bridges/bus owners may be configurable to use only static EIDs.

### 1076 8.17.3 Use of Static Physical Addresses

1077 In many simple topologies, it is desirable to use devices that have statically configured physical  
1078 addresses. This can simplify the implementation of the device. For example, an SMBus device that is not  
1079 used in a hot-plug application would not need to support the SMBus address assignment (SMBus ARP)  
1080 protocol. Fixed addresses can also aid in identifying the location and use of an MCTP device in a system.  
1081 For example, if a system has two otherwise identical MCTP devices, a system vendor will know that the  
1082 device at address "X" is the one at the front of the motherboard, and the device at address "Y" is at the  
1083 back, because that is how they assigned the addresses when the system was designed.

1084 Therefore, MCTP transport bindings, such as for SMBus, are allowed to support devices being at static  
1085 physical addresses without requiring the binding to define a mechanism that enables the bus owner to  
1086 discover MCTP devices that are using static addresses.

1087 In this case, the bridge or bus owner must have a prior knowledge of the addresses of those devices to  
1088 be able to assign EIDs to those devices and to support routing services for those devices. To support this  
1089 requirement, the following requirements and recommendations are given to device vendors:

- 1090 • Devices that act as bus owners or bridges and are intended to support MCTP devices that use  
1091 static physical addresses should provide a non-volatile configuration option that enables the  
1092 system integrator to configure which device addresses are being used for devices on each bus  
1093 that is owned by the bridge/bus owner.
- 1094 • The mechanism by which this non-volatile configuration occurs is specific to the device vendor.  
1095 In many cases, the physical address information will be kept in some type of non-volatile  
1096 storage that is associated with the device and gets loaded when the device is manufactured or  
1097 when the device is integrated into a system. In other cases, this information may be coded into  
1098 a firmware build for the device.

**1099 8.17.4 Endpoint ID Assignment Process for Bus Owners/Bridges**

1100 The bus owner/bridge must get its own EID assignment, and a pool of EIDs, as follows. These steps only  
1101 apply to bus owner/bridge devices that are not the topmost bus owner.

- 1102 • Bus owners/bridges must be pre-configured with non-volatile information that identifies which  
1103 buses they own. (How this configuration is accomplished is device/vendor specific and is  
1104 outside the scope of this specification.)
- 1105 • The bus owner/bridge announces its presence on any buses *that it does not own* to get an EID  
1106 assignment for that bus. The mechanism by which this announcement occurs is dependent on  
1107 the particular physical transport binding and is defined as part of the binding specification.
- 1108 • The bus owner/bridge waits until it gets its own EID assignment for one of those buses through  
1109 the Set Endpoint ID command.
- 1110 • The bus owner/bridge indicates the size of the EID pool it requires by returning that information  
1111 in the response to the Set Endpoint ID command.
- 1112 • For each bus where the bus owner/bridge is itself an "owned" device, the bus owner/bridge will  
1113 be offered a pool of EIDs by being sent an Allocate Endpoint IDs command from the bus owner.
- 1114 • The bus owner/bridge accepts allocations only from the bus of the "first" bus owner that gives it  
1115 the allocation, as described in the Allocate Endpoint IDs command description in 8.10. If it gets  
1116 allocations from other buses, they are rejected.

1117 The bus owner can now begin to build a routing table for each of the buses that it owns, and accept  
1118 routing information update information. Refer to 8.19 for more information.

**1119 8.17.5 Reclaiming EIDs from Hot-Plug Devices**

1120 Bridges will typically have a limited pool of EIDs from which to assign and allocate to devices. (This also  
1121 applies when a single bus owner supports hot-plug devices.) It is important for bridges to reclaim EIDs so  
1122 that when a device is removed, the EID can later be re-assigned when a device is plugged in. Otherwise,  
1123 the EID pool could become depleted as devices are successively removed and added.

1124 EIDs for endpoints that use static addresses are not reclaimed.

1125 No mechanism is specified in the MCTP base protocol for detecting device removal when it occurs.  
1126 Therefore, the general approach to detecting whether a device has been removed is to re-enumerate the  
1127 bus when a new device is added and an EID or EID pool is being assigned to that device.

1128 The following approach can be used to detect removed hot-plug devices: The bus owner/bridge can  
1129 detect a removed device or devices by validating the EIDs that are presently allocated to endpoints that  
1130 are directly on the bus and identifying which EIDs are missing. It can do this by attempting to access each  
1131 endpoint that the bridge has listed in its routing table as being a device that is directly on the particular  
1132 bus. Attempting to access each endpoint can be accomplished by issuing the Get Endpoint ID command  
1133 to the physical address of each device and comparing the returned result to the existing entry in the  
1134 routing table. If there is no response to the command, or if there is a mismatch with the existing routing  
1135 information, the entry should be cleared and the corresponding EID or EID range should be returned to  
1136 the "pool" for re-assignment. The bus owner/bridge can then go through the normal steps for EID  
1137 assignment.

1138 This approach should work for all physical transport bindings, because it keeps the "removed EID"  
1139 detection processing separated from the address assignment process for the bus.

1140 In some cases, a hot-plug endpoint may temporarily go into a state where it does not respond to MCTP  
1141 control messages. Depending on the medium, it is possible that when the endpoint comes back on line, it  
1142 does not request a new EID assignment but instead continues using the EID it had originally assigned. If  
1143 this occurs while the bus owner is validating EIDs to see if any endpoints are no longer accessible, it is

1144 possible that the bus owner will assume that the endpoint was removed and reassign its EID to a newly  
1145 inserted endpoint, unless other steps are taken:

- 1146 • The bus owner must wait at least  $T_{RECLAIM}$  seconds before reassigning a given EID (where  
1147  $T_{RECLAIM}$  is specified in the physical transport binding specification for the medium used to  
1148 access the endpoint).
- 1149 • Reclaimed EIDs must only be reassigned after all unused EIDs in the EID pool have been  
1150 assigned to endpoints. Optionally, additional robustness can be achieved if the bus owner  
1151 maintains a short FIFO list of reclaimed EIDs (and their associated physical addresses) and  
1152 allocates the older EIDs first.
- 1153 • A bus owner shall confirm that an endpoint has been removed by attempting to access it after  
1154  $T_{RECLAIM}$  has expired. It can do this by issuing a Get Endpoint ID command to the endpoint to  
1155 verify that the endpoint is still non-responsive. It is recommended that this be done at least three  
1156 times, with a delay of at least  $1/2 * T_{RECLAIM}$  between tries if possible. If the endpoint continues  
1157 to be non-responsive, it can be assumed that it is safe to return its EID to the pool of EIDs  
1158 available for assignment.

### 1159 **8.17.6 Additional Requirements for Hot-Plug Endpoints**

1160 Devices that are hot-plug must support the Get Endpoint UUID command. The purpose of this  
1161 requirement is to provide a common mechanism for identifying when devices have been changed.

1162 Endpoints that go into states where they temporarily do not respond to MCTP control messages shall re-  
1163 announce themselves and request a new EID assignment if they are "off line" for more than  $T_{RECLAIM}$   
1164 seconds, where  $T_{RECLAIM}$  is specified in the physical transport binding specification for the medium used  
1165 to access the endpoint.

### 1166 **8.17.7 Additional Requirements for Devices with Multiple Endpoints**

1167 A separate EID is utilized for each MCTP bus that a non-bridge device connects to. In many cases, it is  
1168 desirable to be able to identify that the same device is accessible through multiple EIDs.

1169 If an endpoint has multiple physical interfaces (ports), the interfaces can be correlated to the device by  
1170 using the MCTP Get Endpoint UUID command (see 10.5) to retrieve the unique system-wide identifier.

1171 Devices connected to multiple buses must support the Get Endpoint UUID command for each endpoint  
1172 and return a common UUID value across all the endpoints. This is to enable identifying EIDs as belonging  
1173 to the same physical device.

### 1174 **8.18 Handling Reassigned EIDs**

1175 Though unlikely, it is still possible that during the course of operation of an MCTP network, a particular  
1176 EID could get reassigned from one endpoint to another. For example, this could occur if a newly hot-swap  
1177 inserted endpoint device gets assigned an EID that was previously assigned to a device that was  
1178 subsequently removed.

1179 Under this condition, it is possible that the endpoint could receive a message that was intended for the  
1180 previously installed device. This is not considered an issue for MCTP control messages because the  
1181 control messages are typically just used by bus owners and bridges for initializing and maintaining the  
1182 MCTP network. The bus owners and bridges are aware of the EIDs they have assigned to endpoints and  
1183 are thus intrinsically aware of any EID reassignment.

1184 Other endpoints, however, are not explicitly notified of the reassignment of EIDs. Therefore,  
1185 communication that occurs directly from one endpoint to another is subject to the possibility that the EID  
1186 could become assigned to a different device in the middle of communication. This must be protected  
1187 against by protocols specific to the message type being used for the communication.

1188 In general, the approach to protecting against this will be that other message types will require some kind  
 1189 of "session" to be established between the intercommunicating endpoints. By default, devices would not  
 1190 start up with an active session. Thus, if a new device is added and it gets a reassigned EID, it will not  
 1191 have an active session with the other device and the other device will detect this when it tries to  
 1192 communicate.

1193 The act of having a new EID assigned to an existing device should have the same effect. That is, if a  
 1194 device gets a new EID assignment, it would "close" any active sessions for other message types.

1195 The mechanism by which other message types would establish and track communication sessions  
 1196 between devices is not specified in this document. It is up to the specification of the particular message  
 1197 type.

1198 **8.19 MCTP Bridging**

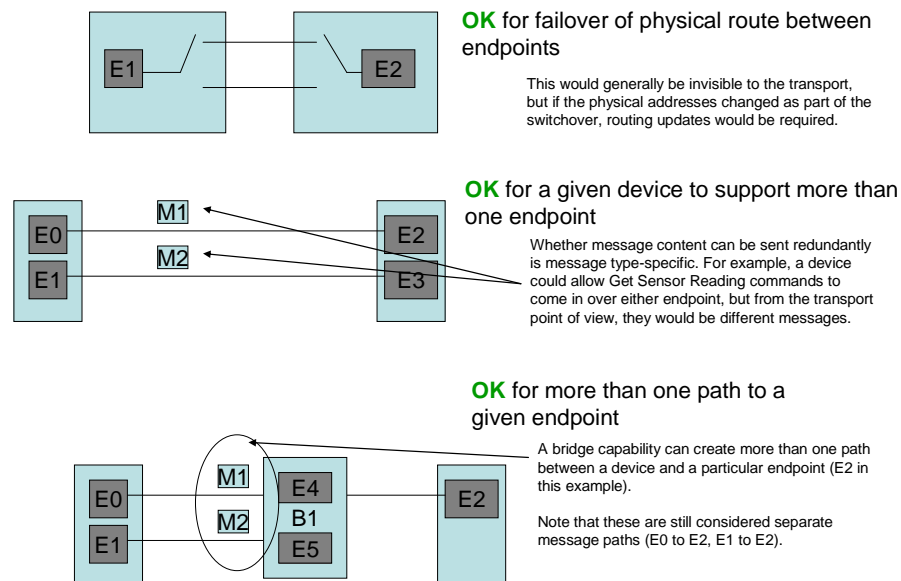
1199 One key capability provided by MCTP is its ability to route messages between multiple buses and  
 1200 between buses of different types. This section describes how routing information is created, maintained,  
 1201 and used by MCTP bridges and MCTP endpoints. Keep the following key points in mind about MCTP  
 1202 bridges:

- 1203 • An MCTP bridge is responsible for routing MCTP packets between at least two buses.
- 1204 • An MCTP bridge is typically the bus owner for at least one of those buses.

1205 **8.19.1 Routing/Bridging Restrictions**

1206 Figure 7 and Figure 8 illustrate some of the supported and unsupported bridging topologies. As shown, it  
 1207 is acceptable for a given topology to have more than one path to get to a given EID. This can occur either  
 1208 because different media are used or because a redundant or failover communication path is desired in an  
 1209 implementation.

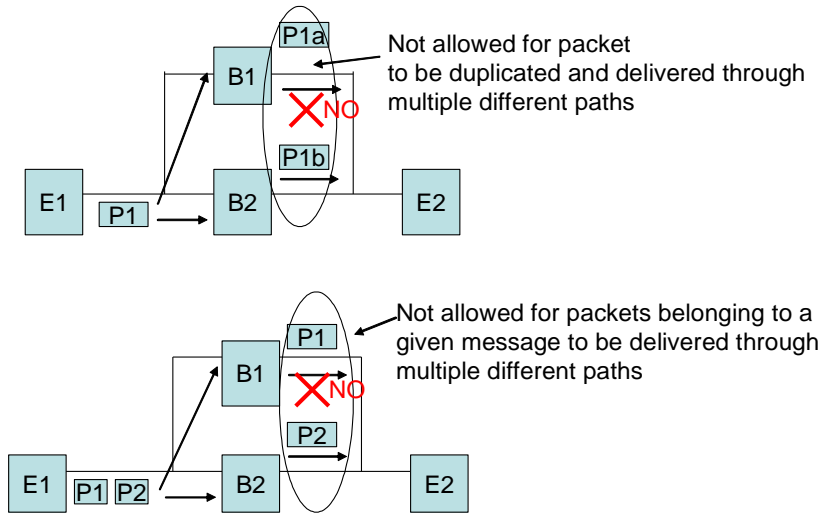
1210 A bridge shall not route or forward packets with a broadcast destination ID.



1211

1212 **Figure 7 – Acceptable Failover/Redundant Communication Topologies**





1213

1214

**Figure 8 – Routing/Bridging Restrictions**

1215 **8.19.2 EID Options for MCTP Bridges**

1216 An MCTP bridge that connects to multiple buses can have a single EID or multiple EIDs through which  
 1217 the bridge's routing configuration and endpoint functionality can be accessed through MCTP control  
 1218 commands. There are three general options:

- 1219 • The bridge uses a single MCTP endpoint
- 1220 • The bridge uses an MCTP endpoint for each bus that connects to a bus owner
- 1221 • The bridge uses an MCTP endpoint for every bus to which it connects

1222 Examples of these different options are shown in Figure 9, and more detailed information on the options  
 1223 is provided following the figure.

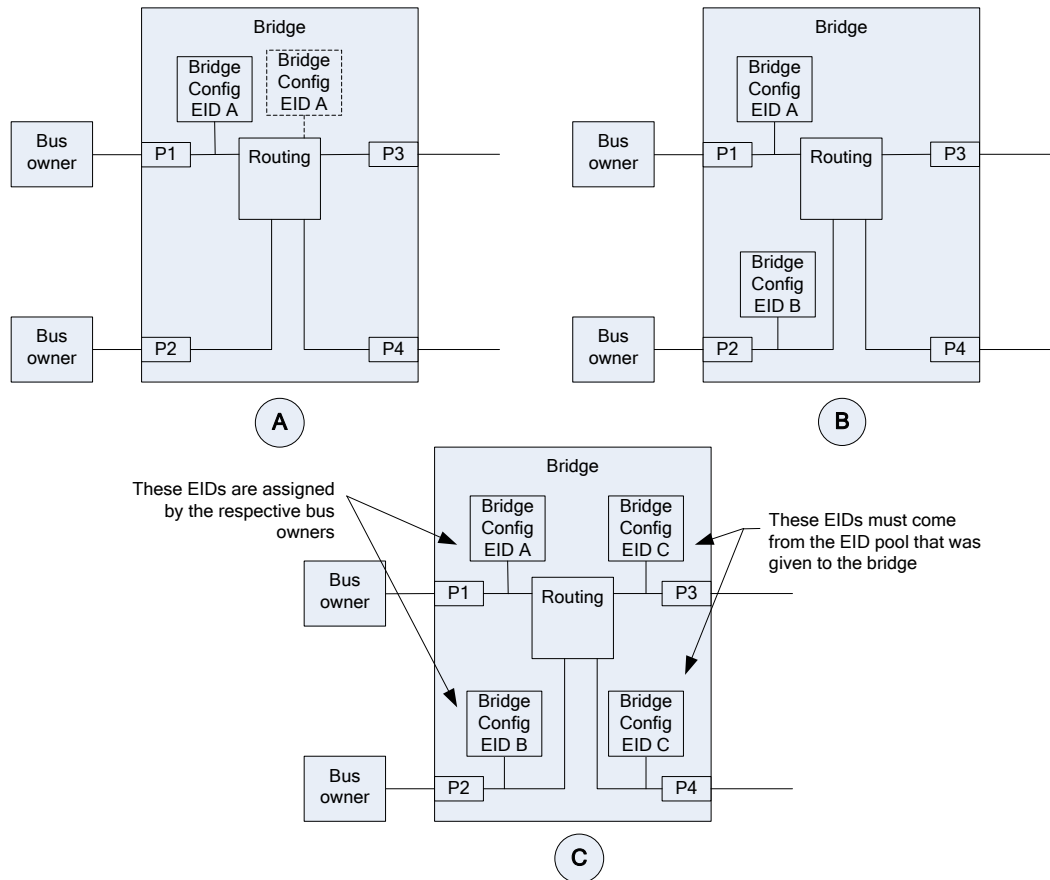


Figure 9 – EID Options for MCTP Bridges

1224

1225

1226 A bridge has only one EID pool. To prevent issues with getting an EID pool allocation from multiple bus  
 1227 owners, a bridge that is accessible through multiple EIDs will only accept EID pool allocation from the first  
 1228 bus that allocation is received from using the Allocate Endpoint IDs command. This behavior is described  
 1229 in more detail in the specification of the Allocate Endpoint IDs command.

1230 If necessary, the Get Endpoint UUID command can be used to correlate that EIDs belong to the same  
 1231 MCTP bridge device. (This correlation is not required for normal initialization and operation of the MCTP  
 1232 network, but it may be useful when debugging.)

1233 The following is a more detailed description of the different EID options for bridges:

1234 • **Single endpoint**

1235 A single endpoint is used to access the bridge's routing configuration and endpoint functionality.  
 1236 Referring to diagram (A) in Figure 9, an implementation may elect to either have the endpoint  
 1237 functionality be directly associated with a particular bus/port (for example, P1) or the  
 1238 functionality can be located on a "virtual bus" that is behind the routing function. In either case,  
 1239 the routing functionality ensures that the EID can be accessed through any of the buses to  
 1240 which the bridge connects.

1241 Although there is a single endpoint, the bridge shall report the need for EID assignment for that  
 1242 endpoint on each bus that is connected to a bus owner (for example, P1, P2). The multiple  
 1243 announcements provide a level of failover capability in the EID assignment process in case a  
 1244 particular bus owner becomes unavailable. The multiple announcements also help support a

1245 consistent EID assignment process across bus owners. To prevent issues with getting  
1246 conflicting EID assignments from multiple bus owners, the bridge will only accept EID pool  
1247 allocation from the first bus that an allocation is received from using the Set Endpoint ID  
1248 command. This behavior is described in more detail in the specification of the Set Endpoint ID  
1249 command. The bridge shall not report the need for EID assignment on any buses that the bridge  
1250 itself owns.

1251 • **Endpoint for each bus connection to a bus owner**

1252 The bridge has one endpoint for each bus connected to a bus owner. This is shown as diagram  
1253 (B) in Figure 9. There are no explicit endpoints associated with buses that are not connected to  
1254 a bus owner (for example, the buses connected to ports P3 and P4, respectively.) Because of  
1255 the way packet routing works, EID A and EID B can be accessed from any of the ports  
1256 connected to the bridge. Thus, the bridge's configuration functionality may be accessed through  
1257 multiple EIDs. Because a separate endpoint communication terminus is associated with each  
1258 port (P1, P2), the bridge can accept an EID assignment for each bus independently.

1259 The bridge shall only report the need for EID assignment on buses that connect to a bus owner,  
1260 and only for the particular MCTP control interface that is associated with the particular bus. For  
1261 example, the bridge would announce the need for EID assignment for the interface associated  
1262 with EID A only through P1, and the need for EID assignment for the interface associated with  
1263 EID B only through P2. The bridge shall not report the need for EID assignment on any buses  
1264 that the bridge itself owns.

1265 • **Endpoint for every bus connection**

1266 The bridge has one endpoint for each bus connected to it, as shown as diagram (C) in Figure 6.  
1267 This includes buses that connect to bus owners (for example, P1, P2) and buses for which the  
1268 bridge is the bus owner (for example, P3, P4). Because of the way packet routing works, any of  
1269 these EIDs can be accessed from any of the ports connected to the bridge.

1270 Because a separate endpoint communication terminus is associated with each owned port (P1,  
1271 P2), the bridge can accept an EID assignment for the bus owners of each bus independently.  
1272 The EIDs associated with the buses that the bridge itself owns (for example, P3, P4) must be  
1273 taken out of the EID pool that is allocated to the bridge.

1274 The bridge shall only report the need for EID assignment on buses that connect to a bus owner,  
1275 and only for the particular MCTP control interface that is associated with the particular bus. For  
1276 example, the bridge would announce the need for EID assignment for the interface associated  
1277 with EID A only through P1, and the need for EID assignment for the interface associated with  
1278 EID B only through P2. The bridge shall not report the need for EID assignment on any buses  
1279 that the bridge itself owns.

### 1280 8.19.3 Routing Table

1281 An MCTP bridge maintains a routing table where each entry in the table associates either a single EID or  
1282 a range of EIDs with a single physical address and bus ID for devices that are on buses that are directly  
1283 connected to the bridge.

1284 If the device is a bridge, there will typically be a range of EIDs that are associated with the physical  
1285 address of the bridge. There may also be an entry with a single EID for the bridge itself.

### 1286 8.19.4 Bridging Process Overview

1287 When a bridge receives an MCTP packet, the following process occurs:

- 1288 1) The bridge checks to see whether the destination EID in the packet matches or falls within the  
1289 range of EIDs in the table.
- 1290 2) If the EID is for the bridge itself, the bridge internally consumes the packet.

- 1291 3) If there is a match with an entry in the routing table, the following steps happen:
- 1292 • The bridge changes the physical addresses in the packet and reformats the medium-
- 1293 specific header and trailer fields as needed for the destination bus.
- 1294 • The destination physical address from the source bus is replaced with the destination
- 1295 physical address for the destination bus obtained from the entry in the routing table.
- 1296 • The bridge replaces the source physical address in the packet it received with the bridge's
- 1297 own physical address on the target bus. This is necessary to enable messages to be
- 1298 routed back to the originator.
- 1299 • Packet-specific transport header and data integrity fields are updated as required by the
- 1300 particular transport binding.
- 1301 4) If there is no match, packets with EID values that are not in the routing table are silently
- 1302 discarded.

### 1303 8.19.5 Endpoint Operation with Bridging

1304 A bridge does not track the packet transmissions between endpoints. It simply takes packets that it

1305 receives and routes them on a per-packet basis based on the destination EID in the packet. It does not

1306 pay attention to message assembly or disassembly or message type-specific semantics, such as

1307 request/response semantics, for packets that it routes to other endpoints.

1308 Most simple MCTP endpoints will never need to know about bridges. Typically, another endpoint will

1309 initiate communication with them. The endpoint can then simply take the physical address and source

1310 EID information from the message and use that to send messages back to the message originator.

1311 An endpoint that needs to originate a "connection" to another MCTP endpoint does need to know what

1312 physical address should be used for messages to be delivered to that endpoint. To get this information, it

1313 needs to query the bus owner for it. An endpoint knows the physical address of the bus owner because it

1314 saved that information when it got its EID assignment.

1315 The Resolve Endpoint ID command requests a bus owner to return the physical address that is to be

1316 used to route packets to a given EID. (This is essentially the MCTP equivalent of the ARP protocol that is

1317 used to translate IP addresses to physical addresses.) The address that is returned in the Resolve

1318 Endpoint ID command response will either be the actual physical address for the device implementing the

1319 endpoint, or it will be the physical address for the bridge to be used to route packets to the desired

1320 endpoint.

1321 Because the physical address format is media-specific, the format of the physical address parameter is

1322 documented in the specifications for the particular media-specific physical transport binding for MCTP (for

1323 example, MCTP over SMBus, MCTP over PCIe, and so on).

1324 If endpoint A has received a message from another endpoint B, it does not need to issue a Resolve

1325 Endpoint ID command. Instead, it can extract the source EID and source physical address from the

1326 earlier message from endpoint B, and then use that as the destination EID and destination physical

1327 address for the message to Endpoint B.

### 1328 8.19.6 Routing Table Entries

1329 Each MCTP device that does bridging must maintain a logical routing table. A bus owner must also

1330 typically maintain a routing table if more than one MCTP device is connected to the bus that it owns. The

1331 routing table is required because the bus owner is also the party responsible for resolving EIDs to

1332 physical addresses.

1333 The internal format that a device uses for organizing the routing table is implementation dependent. From  
 1334 a logical point of view, each entry in a routing table will be comprised of at least three elements: An EID  
 1335 range, a bus identifier, and a bus address. This is illustrated in Figure 10.

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

1336 **Figure 10 – Basic Routing Table Entry Fields**

1337 The *EID range* specifies the set of EIDs that can be reached through a particular bus address on a given  
 1338 bus. Because the bus ID and bus address may correspond to a particular "port" on a bridge, it is possible  
 1339 that there can be multiple non-contiguous ranges (multiple routing table entries) that have the same bus  
 1340 ID/bus address pair route. EIDs and EID ranges can be categorized into three types: downstream,  
 1341 upstream, and local. "Downstream" refers to EIDs that are associated with routing table entries that are  
 1342 for buses that are owned by the bridge that is maintaining the routing table. "Upstream" refers to EIDs that  
 1343 are associated with routing table entries that route to buses that are not owned by the bridge that is  
 1344 maintaining the routing table.

1345 "Local" refers to the EIDs for routing table entries for endpoints that are on buses that are directly  
 1346 connected to the bridge that is maintaining the routing table. A particular characteristic of entries for local  
 1347 EIDs is that the Resolve Endpoint ID command is issued from the same bus that the endpoint is on. The  
 1348 bridge/bus owner delivers the physical address for that endpoint rather than the physical address  
 1349 associated with a routing function. This facilitates allowing endpoints on the same the bus to  
 1350 communicate without having to go through an MCTP routing function.

1351 A routing table entry may not be "local" even if two endpoints are located on the same bus. An implementation may  
 1352 require that different endpoints go through the routing function to intercommunicate even if the endpoints are part of  
 1353 the same bus.

1354 The *bus ID* is an internal identifier that allows the MCTP device to identify the bus that correlates to this  
 1355 route. MCTP does not require particular values to be used for identifying a given physical bus connection  
 1356 on a device. However, this value will typically be a 0-based numeric value.

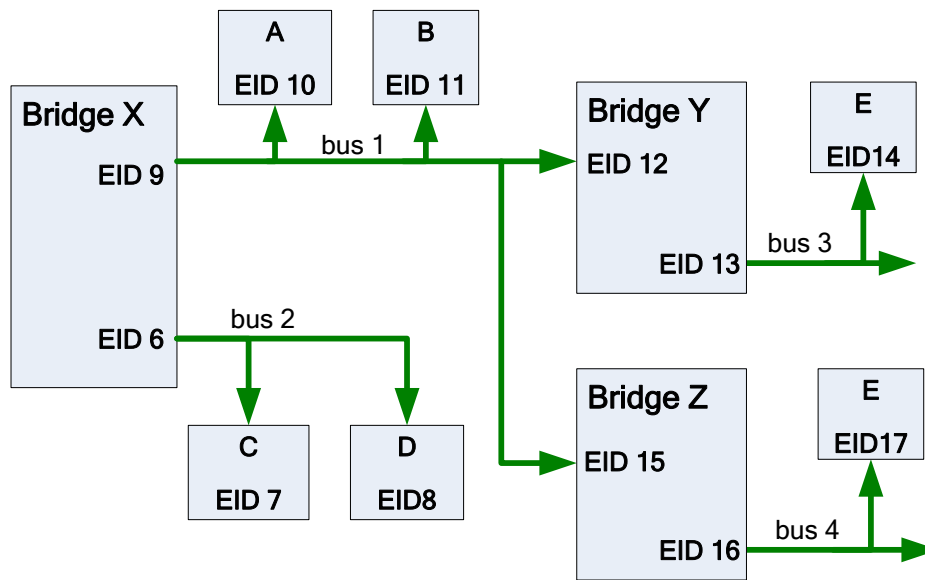
1357 EXAMPLE: A device that had three buses would typically identify them as buses "0", "1", and "2".

1358 The *bus address* is the physical address of a specific device on the bus through which the EIDs specified  
 1359 in the *EID range* can be reached. This can either be the physical address corresponding to the  
 1360 destination endpoint, or it can be the physical address of the next bridge in the path to the device. The  
 1361 format of this address is specific to the particular physical medium and is defined by the physical medium  
 1362 transport binding.

### 1363 **8.19.7 Routing Table Creation**

1364 This section illustrates the types of routing information that a bridge requires, and where the information  
 1365 comes from. This section also describes the steps that a bus owner must use to convey that information  
 1366 for a given bus.

1367 Figure 11 helps illustrate the steps that are required to completely establish the routing information  
 1368 required by a bridge (bridge Y). The arrows in Figure 11 point outward from the bus owner and inward to  
 1369 "owned" endpoints on the bus.



1370

1371

Figure 11 – Routing Table Population

### 1372 8.19.7.1 Routing Table Population Example

1373 With reference to Figure 11, the following items describe the information that bridge Y will need for routing  
1374 messages in the example topology shown:

- 1375 • It needs a set of EIDs allocated to it to use for itself and to allocate to other devices (for  
1376 example, EIDs 12:14). These are allocated to it by the bus owner (bridge X).
- 1377 • It needs a routing table that has an entry that maps EID 14 to the physical address for device E  
1378 on bus 3.
- 1379 • It needs routing table entries for the local devices on bus 1, which are: bridge X (EID 9), device  
1380 A (EID 10), device B (EID 11), and bridge Z (EID 15), assuming that devices A and B are to be  
1381 reached by bridge Y without having to go through bridge X. This information must be given to it  
1382 by the bus owner (bridge X).
- 1383 • It needs to know that EIDs 6:8 are accessed through bus owner/bridge X. Therefore, it needs a  
1384 routing table entry that maps the EID range 6:8 to the physical address for bridge X on bus 1.  
1385 This information must also be given to it by the bus owner (bridge X).
- 1386 • It needs to know that EIDs 16:17 are accessed through bridge Z. Therefore, it needs a routing  
1387 table entry that maps the EID range 16:17 to the physical address for bridge Z on bus 1.  
1388 Because the bus owner (bridge X) allocated that range of EIDs to bridge Z in the first place, this  
1389 information is also given to bridge Y by the bus owner (bridge X).

### 1390 8.19.7.2 Bus Initialization Example

1391 Starting with the description of what bridge Y requires, the following task list shows the steps that bridge  
1392 X must take to provide routing information for bus 1. Bridge X must:

- 1393 1) Assign EIDs to devices A, B, C, D, bridge Y, and bridge Z. This is done using the Set Endpoint  
1394 ID command. The response of the Set Endpoint ID command also indicates whether a device  
1395 wants an additional pool of EIDs.

- 1396 2) Allocate EID pools to bridge Y and bridge Z. This is done using the Allocate Endpoint IDs  
1397 command.
- 1398 3) Tell bridge Y the physical addresses and EIDs for devices A and B, bridge X (itself), and bridge  
1399 Z on bus 1. This is done using the Routing Information Update command.
- 1400 4) Tell bridge Y that EIDs 16:17 are accessed through the physical address for bridge Z on bus 1.  
1401 This is also done using the Routing Information Update command. (Steps 3 and 4 can be  
1402 combined and covered with one instance of the command.)
- 1403 5) Tell bridge Z the physical addresses and EIDs for devices A and B, bridge X (itself), and bridge  
1404 Y on bus 1. This is also done using the Routing Information Update command.
- 1405 6) Tell bridge Z that EIDs 13:14 are accessed through the physical address for bridge Y on bus 1.  
1406 This is also done using the Routing Information Update command. (Steps 5 and 6 can be  
1407 combined and covered with one instance of the command.)
- 1408 7) Tell bridge Y and bridge Z that EIDs 6:8 are accessed through bridge X on bus 1. This is also  
1409 done using the Routing Information Update command. This step could also be combined with  
1410 steps 3 and 4 for bridge Y and steps 5 and 6 for bridge Z.

### 1411 8.19.8 Routing Table Updates Responsibility for Bus Owners

1412 After it is initialized for all bridges, routing table information does not typically require updating during  
1413 operation. However, updating may be required if a bridge is added as a hot-plug device. In this case,  
1414 when the bridge is added to the system, it will trigger the need for the bus owner to assign it an EID,  
1415 which will subsequently cause the request for EID pool allocations, and so on. At this time, the bus owner  
1416 can simply elect to re-run the steps for bus initialization as described in 8.19.7.2.

### 1417 8.19.9 Consolidating Routing Table Entries

1418 MCTP requires that when an EID pool is allocated to a device, the range of EIDs is contiguous and  
1419 follows the EID for the bridge itself. Thus, a bridge can elect to consolidate routing table information into  
1420 one entry when it recognizes that it has received an EID or EID range that is contiguous with an existing  
1421 entry for the same physical address and bus. (The reason that EID allocation and routing information  
1422 updates are not done as one range using the same command is because of the possibility that a device  
1423 may have already received an allocation from a different bus owner.)

## 1424 8.20 Bridge and Routing Table Examples

1425 The following examples illustrate different bridge and MCTP network configurations and the  
1426 corresponding information that must be retained by the bridge for MCTP packet routing and to support  
1427 commands such as Resolve Endpoint ID and Query Hop.

1428 The following clauses (including Table 4 through Table 6) illustrate possible topologies and ways to  
1429 organize the information that the bridge retains. Implementations may elect to organize and store the  
1430 same information in different ways. The important aspect of the examples is to show what information is  
1431 kept for each EID, to show what actions cause an entry to be created, and to show how an EID or EID  
1432 range can in some cases map to more than one physical address.

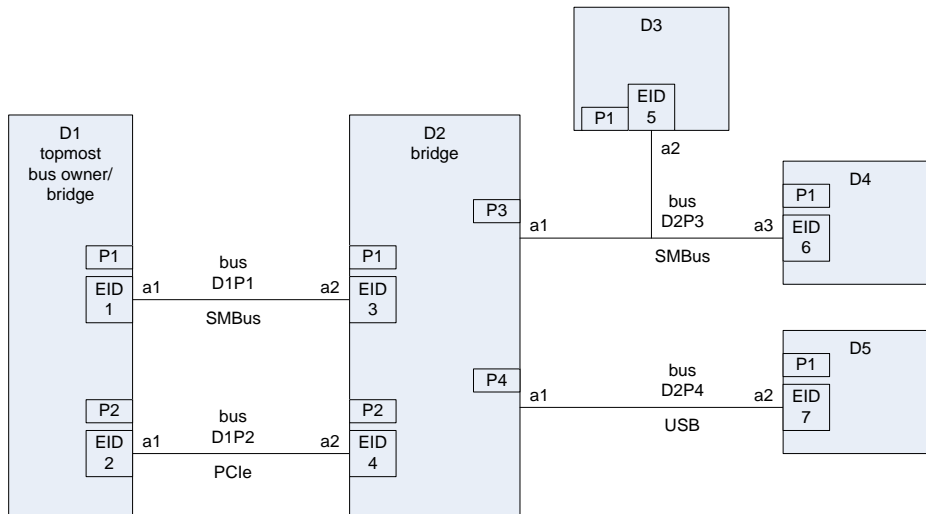
1433 The examples show a possible time order in which the entries of the table are created. Note that a given  
1434 implementation of the same example topology could have the entries populated in a different order. For  
1435 example, if there are two bus owners connected to a bridge, there is no fixed order that the bus owners  
1436 would be required to initialize a downstream bridge. Additionally, there is no requirement that bus owners  
1437 perform EID assignment or EID pool allocation in a particular order. One implementation may elect to  
1438 allocate EID pools to individual bridges right after it has assigned the bridge its EID. Another  
1439 implementation may elect to assign all the EIDs to devices first, and then allocate the EID pools to  
1440 bridges.

1441 **8.20.1 Example 1: Bridge D2 with an EID per "Owned" Port**

1442 Figure 12 shows the routing table in a bridge (D2), where D2 has an EID associated with each bus  
 1443 connected to a bus owner. In this example, D1 is not implementing any internal bridging between its P1  
 1444 and P2. Consequently, EID2 cannot be reached by bridging through EID1 and vice versa (see Table 4).

1445 NOTE: If there was internal bridging, D1 would need to provide routing information that indicated that EID2 was  
 1446 reachable by going through EID1 and vice versa. In this case, D1 would provide routing information that EID range  
 1447 (EID1...EID2) would be accessed through D1P1a1 on SMBus and D1P1a2 on PCIe.

1448 **Key: D = device, P = port, a = physical address**



1449

1450 **Figure 12 – Example 1 Routing Topology**

1450

1451 **Table 4 – Example 1 Routing Table for D2**

1451

Time	EID	EID Access Port	Medium Type	Access Physical Address	Device/Entry Type	Entry Was Created and Populated By
↓	EID 3	P1	SMBus	D1P1a2	Bridge, Self	Self when EID was assigned by D1
	EID 4	P2	PCIe	D1P2a2	Bridge, Self	Self when EID was assigned by D1
	EID 5	P3	SMBus	D2P3a2	Endpoint	Self after D1 assigned EID pool (typically the entry will not be created until after the bridge D2 assigns EID 5 to D3)
	EID 6	P3	SMBus	D2P3a3	Endpoint	Self after D1 assigned EID pool (typically the entry will not be created until after the bridge D2 assigns EID 6 to D4)
	EID 7	P4	USB	D2P4a2	Endpoint	Self after D1 assigned EID pool (typically the entry will not be created until after the bridge D2 assigns EID 7 to D5)
	EID 1	P1	SMBus	D1P1a1	Bridge	D1 through Routing Information Update command
	EID 2	P2	PCIe	D1P2a1	Bridge	D1 through Routing Information Update command



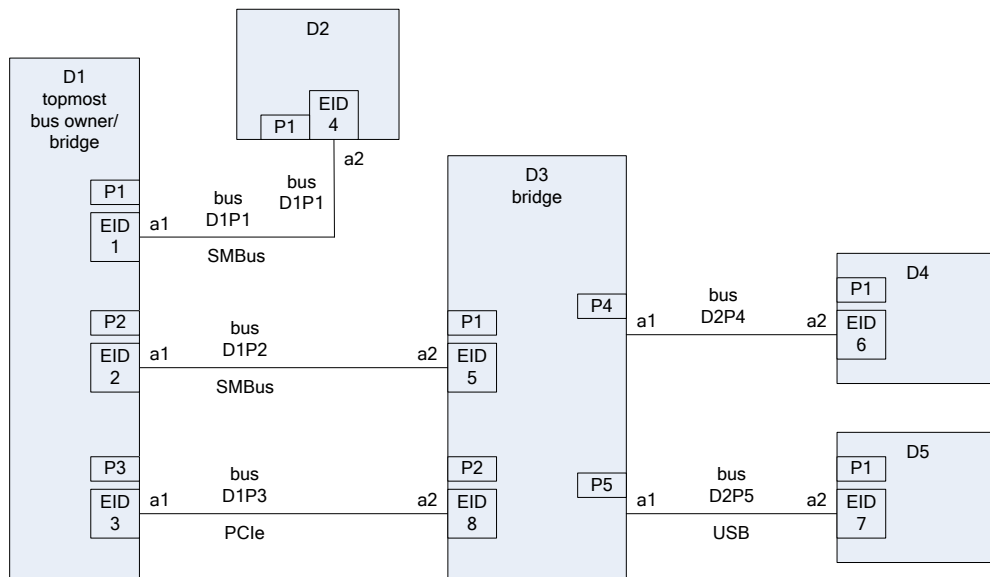
1452 **8.20.2 Example 2: Topmost Bus Owner D1**

1453 Figure 13 assumes the following:

- 1454 • D1 assigns its internal EIDs first.
- 1455 • The buses are handled in the order D1P1, D1P2, D1P3.
- 1456 • D1 allocates the EID pool to bridges right after it has assigned the EID to the device.

1457 Similar to Example 1, this example assumes that there is no internal bridging within D1 between P1, P2,  
 1458 and P3. This scenario is reflected in Table 5.

1459 **Key: D = device, P = port, a = physical address**



1460

1461

**Figure 13 – Example 2 Routing Topology**

1462

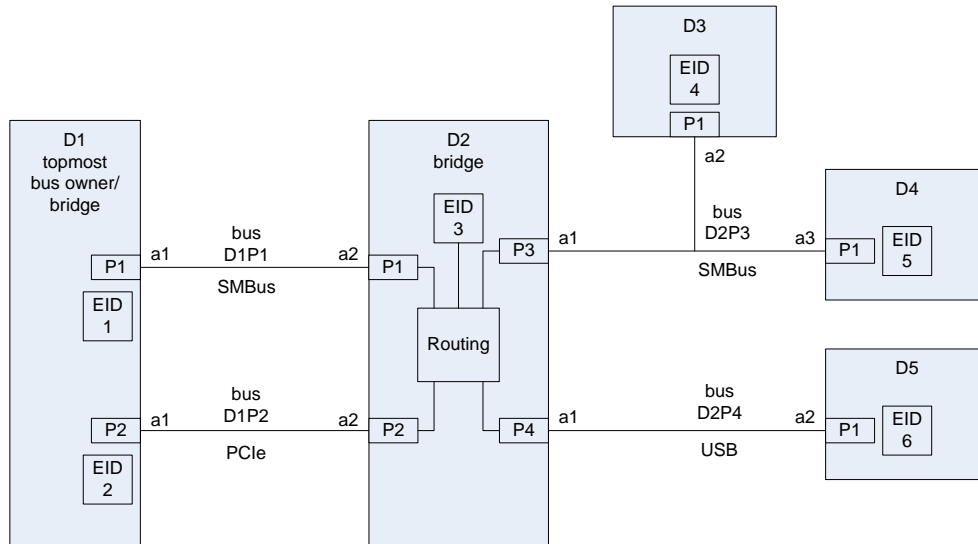
**Table 5 – Example 2 Routing Table for D1**

EID	EID Access Port	Medium Type	Access Physical Address	Device/Entry Type	Entry Was Created and Populated By
EID 1	P1	SMBus	D1P1a1	Bridge, self	Self
EID 2	P2	SMBus	D1P2a1	Bridge, self	Self
EID 3	P3	PCIe	D1P3a1	Bridge, self	Self
EID 4	P1	SMBus	D1P1a2	Endpoint	Self upon assigning EID to device D2
EID 5	P2	SMBus	D1P2a2	Bridge	Self upon assigning EID 5 to bridge D3
EID 6:7	P2	SMBus	D1P2a2	Bridge pool	Self upon assigning EID pool to bridge D3
EID 8	P3	PCIe	D1P3a2	Bridge	Self upon assigning EID 8 to bridge D3
EID 6:7	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

1463 **8.20.3 Example 3: Bridge D2 with Single EID**

1464 Figure 14 assumes that bridge D2 has a single EID and gets its EID assignment and EID allocation  
 1465 through bus D1P1 first, and that bus D1P2 later gets initialized. This scenario is reflected in Table 6.

1466 **Key: D = device, P = port, a = physical address**



1467

1468

**Figure 14 – Example 3 Routing Topology**

1469

**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 3	P1	D1P1a2	Bridge, self	All four entries created by self (bridge) upon receiving initial EID assignment from D1 through P1
EID 3	P2	D1P2a2	Bridge, self	
EID 3	P3	D2P3a1	Bridge, self	
EID 3	P4	D2P4a1	Bridge, self	
EID 4	P3	D2P3a2	Endpoint	Self after D1 allocated EID pool (typically the entry will not be created until after the bridge D2 assigns EID 4 to D3)
EID 5	P3	D2P3a3	Endpoint	Self after D1 allocated EID pool (typically the entry will not be created until after the bridge D2 assigns EID 5 to D4)
EID 6	P3	D2P4a2	Endpoint	Self after D1 allocated EID pool (typically the entry will not be created until after the bridge D2 assigns EID 6 to D5)
EID 1:2	P1	D1P1a1	Bridge	D1 through Routing Information Update command
EID 1:2	P2	D1P2a1	Bridge	D1 through Routing Information Update command

1470 **8.20.4 Additional Information Tracked by Bridges**

1471 In addition to the information required to route messages between different ports, a bridge has to track  
 1472 information to handle MCTP control commands related to the configuration and operation of bridging  
 1473 (shown in Table 7).

1474 **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.

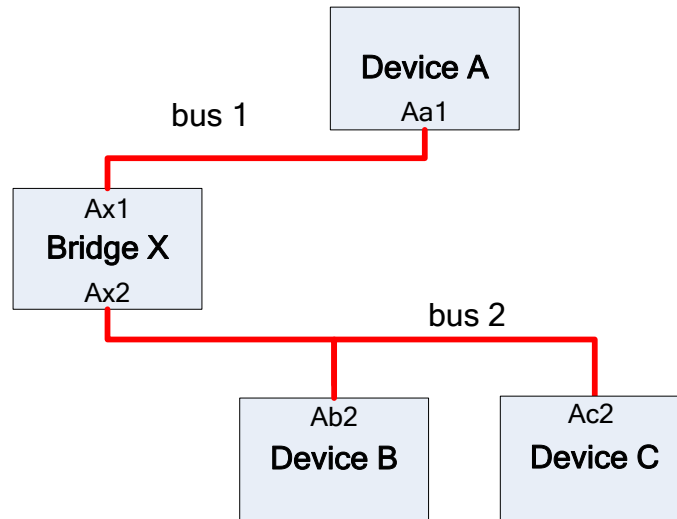
1475 **8.21 Endpoint ID Resolution**

1476 When a device uses the Resolve Endpoint ID command to request the resolution of a given endpoint to a  
 1477 physical address, the bridge must respond based on which bus the request came in on.

1478 For example, consider Figure 15. If device A wishes to get the physical address needed to send a  
 1479 message to device C, it sends a Resolve Endpoint ID command to bus owner bridge X through address  
 1480 Ax1. Because device A must go through bridge X to get to device C, bridge X responds with its physical  
 1481 address Ax1.

1482 When device B wishes to know the address to use to communicate with device C, it sends a Resolve  
 1483 Endpoint ID request to bridge X through address Ax2. In this case, bridge X can respond by giving device  
 1484 B the direct physical address of device C on bus 2, Ac2.

1485 Thus, the Resolve Endpoint ID command can return a different response based on the bus from which  
 1486 the Resolve Endpoint ID command was received.



notation:

Ab2 = physical Address of device b on bus 2.

1487

1488

**Figure 15 – Endpoint ID Resolution**

1489

### 8.21.1 Resolving Multiple Paths

1490

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

1491

1492

1493

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

1494

1495

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

1496

1497

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

1498

1499

1500

1501

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

1502

1503

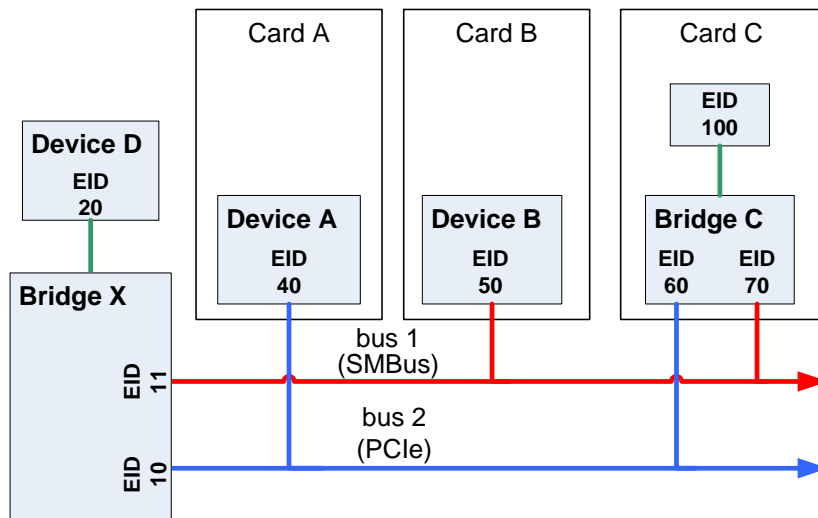


Figure 16 – Resolving Multiple Paths

1504

1505

## 8.22 Bridge and Bus Owner Implementation Recommendations

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

### 8.22.1 Endpoint ID Pool Recommendations

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

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

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

### 8.22.2 Routing Table Size Recommendations

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

- **PCIe slots**

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

This means providing support for at least four directly connected endpoints per card. (Other endpoints may be behind bridges on the card, but this does not affect the routing table size for the bus owner.) This implies at least four routing table entries per PCIe slot. Thus, a device that

- 1530 was designed to support system implementations with eight PCIe slots should have support for  
1531 32 routing table entries.
- 1532 • **Planar PCIe devices**
- 1533 In most PC systems, PCIe would be typically implemented as a single MCTP bus owned by a  
1534 single device as the bus owner. Thus, the number of static devices should be proportional to the  
1535 number of PCIe devices that are built into the motherboard.
- 1536 Typically, this is fewer than eight devices. Thus it is recommended to support at least eight  
1537 entries for static PCIe devices.
- 1538 • **Static SMBus MCTP devices**
- 1539 The routing table should also be sized to support an additional number of "static" devices on  
1540 owned buses. At this time, it is considered unlikely that more than a few MCTP devices would  
1541 be used on a given SMBus. Most devices would be non-intelligent sensor and I/O devices  
1542 instead. Conservatively, it is recommended that at least four entries be provided for each  
1543 SMBus that the device owns.
- 1544 Example 1: "client" capable device
- |      |                                   |                            |
|------|-----------------------------------|----------------------------|
| 1545 | Four PCIe slots                   | → 16 routing table entries |
| 1546 | Two owned SMBus                   | → +8 entries               |
| 1547 | <u>Static PCIe device support</u> | <u>→ +8 entries</u>        |
| 1548 |                                   | ~32 entries or more        |
- 1549 Example 2: volume server capable
- |      |                                   |                            |
|------|-----------------------------------|----------------------------|
| 1550 | Eight PCIe slots                  | → 32 routing table entries |
| 1551 | Four owned SMBus                  | → +16 entries              |
| 1552 | <u>Static PCIe device support</u> | <u>→ +8 entries</u>        |
| 1553 |                                   | ~56 entries or more        |

## 1554 8.23 Path and Transmission Unit Discovery

1555 The transmission unit is defined as the size of the MCTP packet payload that is supported for use in  
1556 MCTP message assembly for a given message. The supported transmission unit sizes are allowed to  
1557 vary on a per-message type basis.

1558 Intermediate bridges and physical media can limit the transmission unit sizes between endpoints.  
1559 Therefore, the MCTP control protocol specifies a mechanism for discovering the transmission unit support  
1560 for the path between endpoints when one or more bridges exist in the path between the endpoints.

1561 The mechanism for path transmission unit discovery also enables the discovery of the bridges and  
1562 number of "hops" that are used to route an MCTP packet from one endpoint to another.

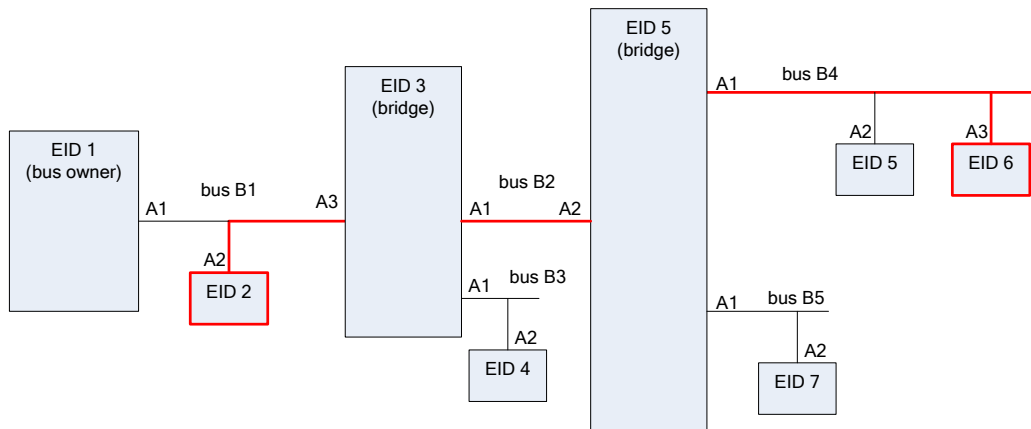
### 1563 8.23.1 Path Transmission Unit Negotiation

1564 The MCTP control protocol only specifies how to discover what the path transmission unit size is for the  
1565 path between endpoints. The MCTP control protocol does not specify a generic mechanism for  
1566 discovering what transmission unit sizes a particular endpoint supports for a given message type.  
1567 Discovery and negotiation of transmission unit sizes for endpoints, if supported, is specified by the  
1568 definition of the particular message type.

### 1569 8.23.2 Path Transmission Unit Discovery Process Overview

1570 This section describes the process used for path transmission unit discovery. The discovery process  
1571 described here is designed to enable one endpoint to discover the path and transmission unit support for

1572 accessing a particular "target" endpoint. It does not define a general mechanism for enabling an endpoint  
 1573 to discover the path between any two arbitrary endpoints. For example, referring to  
 1574 Figure 17, the process defines a way for the endpoint at EID 2 to discover the path/transmission unit  
 1575 support on the route to endpoint at EID 6, but this process does not define a process for EID 2 to discover  
 1576 the path/transmission unit support between EID 4 and EID 6.



1577

1578

**Figure 17 – Example Path Routing Topology**

1579 The following example provides an overview of the path/transmission unit discovery process. The  
 1580 example presumes that the MCTP network has already been initialized. Referring to  
 1581 Figure 17, the endpoint with EID 2 wishes to discover the path used to access the endpoint with EID 6.  
 1582 This discovery is accomplished using just two commands, Resolve Endpoint ID and Query Hop, as  
 1583 follows:

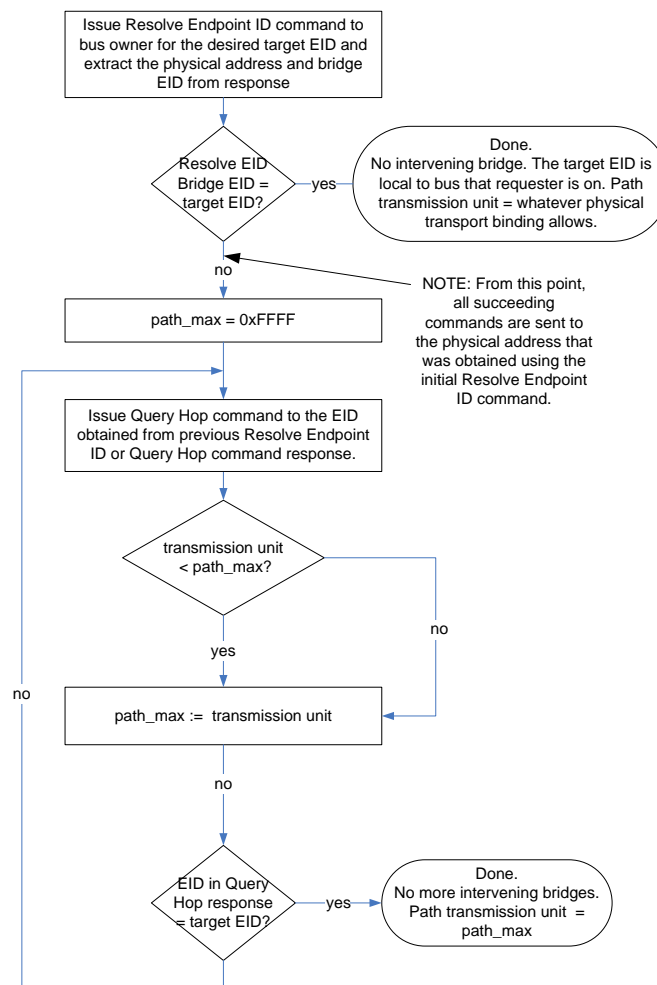
- 1584 1) EID 2 first issues a Resolve Endpoint ID command to the bus owner, EID 1, with EID 6 as the  
 1585 EID to resolve.
- 1586 2) EID 1 returns the physical address and EID of the bridge, EID 3 in the Resolve Endpoint ID  
 1587 command response.
- 1588 3) EID 2 queries the bridge, EID 3, using a Query Hop command with EID 6 (the "target" EID) as  
 1589 the request parameter. Note that EID 2 does not need to do another Resolve Endpoint ID  
 1590 command because it already received the physical address of EID 3 from the original Resolve  
 1591 Endpoint ID command.
- 1592 4) Bridge EID 3 responds to the Query Hop command by returning EID 5, which is the EID of the  
 1593 next bridge required to access EID 6. The bridge EID 3 also returns the transmission unit  
 1594 support that it offers for routing to the target EID.
- 1595 5) EID 2 then sends a Query Hop command to the bridge at EID 5. Note that EID 2 does not need  
 1596 to do another Resolve Endpoint ID command because it already received the physical address  
 1597 of EID 5 from the original Resolve Endpoint ID command.
- 1598 6) Bridge EID 5 responds to the Query Hop command by returning EID 6, which, because it is the  
 1599 EID of the target endpoint, tells EID2 that bridge EID 5 was the last "hop" in the path to EID 6.  
 1600 The bridge EID 5 also returns the transmission unit support that it offers for routing to the target  
 1601 EID.
- 1602 7) At this point, the bridges in the path to EID 6 have subsequently been discovered and their  
 1603 respective transmission unit support returned. The effective transmission unit support for the

1604 path to EID 6 will be the lesser of the transmission unit support values returned by the two  
 1605 bridges.

1606 **8.23.3 Path Transmission Unit Discovery Process Flowchart**

1607 The following flowchart (Figure 15) shows a generic algorithm for discovering the bridges in the path from  
 1608 one endpoint to a given target endpoint and the path transmission unit support. The flowchart has been  
 1609 intentionally simplified. Note that while the Query Hop command actually supports returning separate  
 1610 transmission unit sizes for the transmit and receive paths, the flowchart is simplified for illustration  
 1611 purposes and just refers to a single transmission unit for both transmit and receive.

1612 Additionally, Figure 18 does not show any explicit steps for error handling nor the process of handling  
 1613 command retries. In general, errors are most likely due to either an invalid EID being sent to the bridge  
 1614 (perhaps due to a programming error at the requester) or the EID not being present in the bridge's routing  
 1615 table. The latter condition could occur under normal operation if the requester did not realize that a  
 1616 routing table update had occurred because of a hot-plug update, for example. This error condition would  
 1617 be indicated by the bridge responding with an `ERROR_INVALID_DATA` completion code.



1618

1619 **Figure 18 – Path Transmission Unit Discovery Flowchart**



1620 **8.24 Path Transmission Unit Requirements for Bridges**

1621 An MCTP bridge routes packets between different buses, but it does not typically interpret the packet  
 1622 payload contents nor does it do assembly of those packets. Exceptions to this are when the bridge is  
 1623 handling packets addressed to its own EID, receives a Broadcast EID, and if the bridge supports different  
 1624 transmission units based on message type. See Table 32 for more information.

1625 **9 MCTP Control Protocol**

1626 MCTP control messages are used for the setup and initialization of MCTP communications within an  
 1627 MCTP network. This section defines the protocol and formatting used for MCTP control messages over  
 1628 MCTP.

1629 **9.1 Terminology**

1630 The terms shown in Table 8 are used when describing the MCTP control protocol.

1631 **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.

1632 **9.1.1 Control Message Classes**

1633 The different types of messages shown in Table 9 are used under the MCTP control message type.

1634 **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.

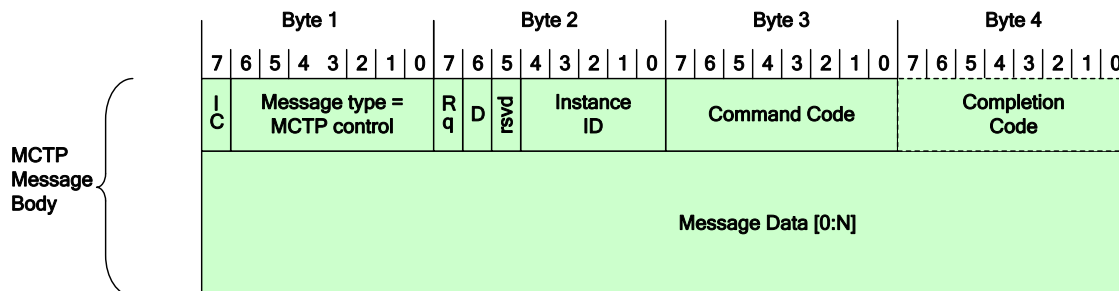
Type	Description
Datagram	Datagrams are "unacknowledged" messages (that is, Datagrams do not have corresponding Response messages). This class of MCTP control message is used to transfer messages when an MCTP control Response message is neither required nor desirable.
Broadcast Request	A broadcast message 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).

1635 **9.2 MCTP Control Message Format**

1636 MCTP control messages use the MCTP control message type (see Table 3). Any message sent with this  
 1637 message type will correspond to the definitions set forth in this section. The basic format of an MCTP  
 1638 control message is shown in Figure 19. Note that the byte offsets shown in Figure 19 are relative to the  
 1639 start of the MCTP message body rather than the start of the physical packet.

1640 **9.2.1 Use of Message Integrity Check**

1641 MCTP control messages do not use a Message Integrity Check field. Therefore, the IC bit in MCTP  
 1642 control messages shall always be 0b.



1643

1644 **Figure 19 – MCTP Control Message Format**

1645 **9.3 MCTP Control Message Fields**

1646 Table 10 lists the common fields for MCTP control messages.

1647 **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 9.5.

Field Name	Description
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 9.5.
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 10. The Command Code that is sent in a Request must be returned in the corresponding Response.
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.	

1648 **9.4 MCTP Control Message Transmission Unit Size**

1649 All MCTP control messages are required to have a packet payload that is no larger than the baseline  
1650 transmission unit size of 64 bytes.

1651 MCTP control messages are carried in a single MCTP packet. Multiple messages are used if an operation  
1652 requires more data to be transferred than can be carried in a single message.

1653 **9.5 Tag Owner (TO), Request (Rq), and Datagram (D) Bit Usage**

1654 For MCTP control messages, the Rq bit shall be set to 1b if the message is a "command" or Request  
1655 message and 0b if the message is a Response message. For Datagram and Broadcast messages, the  
1656 Rq bit shall always be set to 1b. MCTP Control messages that have unexpected or incorrect flag bit  
1657 values shall be silently discarded by the receiver of the message.

1658 For the present specification, Requests and Datagrams are only issued from tag owners ( TO bit = 1b).  
1659 Provision has been left for the definition of possible future Datagrams that are not issued from tag owners  
1660 (see Table 11).

1661 **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

MCTP Control Message Class	Destination EID Value	Tag Owner (TO) bit	Request (Rq) bit	Datagram (D) bit
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			

## 1662 9.6 Concurrent Command Processing

1663 This section describes the specifications and requirements for handling concurrent overlapping MCTP  
1664 control requests by endpoints.

### 1665 9.6.1 Requirements for Responders

1666 An endpoint is not required to process more than one request at a time (that is, it can be "single threaded"  
1667 and does not have to accept and act on new requests until it has finished responding to any previous  
1668 request).

1669 A responder that is not ready to accept a new request can either silently discard the request, or it can  
1670 respond with an `ERROR_NOT_READY` message completion code.

1671 A responder that can accept and process more than one request at a time is not required to return  
1672 responses in the order that the requests were received.

### 1673 9.6.2 Requirements for Requesters

1674 An endpoint that issues MCTP control Requests to another endpoint must wait until it gets the response  
1675 to the particular request, or times out waiting for the response, before issuing a new request, Datagram,  
1676 or Broadcast Datagram.

1677 An endpoint that issues MCTP control Requests is allowed to have multiple requests outstanding  
1678 simultaneously to *different* responder endpoints.

1679 An endpoint that issues MCTP control Requests should be prepared to handle responses that may not  
1680 match the request (that is, it should not automatically assume that a response that it receives is for a  
1681 particular request). It should check to see that the command code and source EID values in the response  
1682 match up with a corresponding outstanding command before acting on any parameters returned in the  
1683 response.

### 1684 9.6.3 Additional Requirements for Bridges

1685 The packets that are routed *through* a bridge's routing functionality are not interpreted by the bridge and  
1686 therefore are not considered to constitute concurrent requests.

1687 A bridge must support at least one outstanding MCTP control request for each bus connection (port)  
 1688 through which MCTP control messages can be used to access the bridge's configuration and control  
 1689 functionality.

1690 Bridges must retain temporal ordering of packets forwarded from one message terminus to another.

## 1691 10 MCTP Control Messages

1692 This section contains detailed descriptions for each MCTP control message. The byte offsets for the  
 1693 Request and Response parameter information given in the tables for the commands indicates the byte  
 1694 offset for the message data starting with the byte following the Command field.

### 1695 10.1 MCTP Control Message Command Codes

1696 Table 12 lists the MCTP control messages and their corresponding command code values. The  
 1697 commands and their associated parameters are specified later in this section. For bridges, the  
 1698 requirements apply equally to all endpoints within the bridge device that are used to configure and control  
 1699 the bridges routing functionality.

1700

**Table 12 – MCTP Control Command Numbers**

Command Code	Command Name	General Description	OMC		Section
			E	B	
0x00	Reserved	Reserved	–	–	–
0x01	Set Endpoint ID	Assigns an EID to the endpoint at the given physical address	Ma Ng	Ca <sup>1</sup> Mg	10.3
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	10.4
0x03	Get Endpoint UUID	Retrieves a per-device unique UUID associated with the endpoint	Ca <sup>2</sup> Og	Ca <sup>2</sup> Og	10.5
0x04	Get MCTP Version Support	Lists which versions of the MCTP control protocol are supported on an endpoint	Ma Og	Ma Og <sup>5</sup>	10.6
0x05	Get Message Type Support	Lists the message types that an endpoint supports	Ma Og	Ma Og	10.7
0x06	Get Vendor Defined Message Support	Used to discover an MCTP endpoint's vendor-specific MCTP extensions and capabilities	Oa Og	Oa Og	10.8
0x07	Resolve Endpoint ID	Used to get the physical address associated with a given EID	Na Og	Ma Og	10.9
0x08	Allocate Endpoint IDs	Used by the bus owner to allocate a pool of EIDs to an MCTP bridge	Na Ng	Ma <sup>6</sup> Mg <sup>6</sup>	10.10
0x09	Routing Information Update	Used by the bus owner to extend or update the routing information that is maintained by an MCTP bridge	Oa Og	Ma <sup>4</sup> Mg <sup>4</sup>	10.11
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	10.12
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 <sup>3</sup> Ng	Ca <sup>3</sup> Cg <sup>3</sup>	10.13

Command Code	Command Name	General Description	OMC		Section
			E	B	
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 <sup>3</sup> Cg <sup>3</sup>	Ca <sup>3</sup> Cg <sup>3</sup>	10.14
0x0D	Discovery Notify	Used to notify the bus owner that an MCTP device has become available on the bus	Na Cg <sup>3</sup>	Ca <sup>3</sup> Cg <sup>3</sup>	10.15
0x0E	Get Network ID	Used to get the MCTP network ID	Ca <sup>7</sup>	Ca <sup>7</sup>	10.16
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	10.17
0x10	Resolve UUID	Used by endpoints to find another endpoint matching an endpoint that uses a specific UUID.	Na Og	Oa Og	10.18
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 many 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 section 10.17.	-	-	10.19

**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 must be accepted by the endpoint and a response generated per the following command descriptions.
- Mg = mandatory to generate. The endpoint must 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 must 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 are required to support the Get Endpoint UUID command.
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 must be

Command Code	Command Name	General Description	OMC		Section
			E	B	
generated by the endpoint if the configuration requires the endpoint to support allocating EID pools to downstream bridges.					
7. See Section 8.19 MCTP Network IDs for information for implementation requirements of this command.					

1701 **10.2 MCTP Control Message Completion Codes**

1702 The command/result code field is used to return management operation results for response messages. If  
 1703 a SUCCESS completion code is returned then the specified response parameters (if any) must also be  
 1704 returned in the response. If an error completion code (not SUCCESS) is returned by the responder, unless  
 1705 otherwise specified, the responder shall not return any additional parametric data and the requester shall  
 1706 ignore any additional parameter data provided in the response (if any). See Table 13 for the completion  
 1707 codes.

1708 **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 must 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

1709 **10.3 Set Endpoint ID**

1710 The Set Endpoint ID command assigns an EID to an endpoint. This command should only be issued by a  
 1711 bus owner to assign an EID to an endpoint at a particular physical address. Since it is assumed the  
 1712 Endpoint does not already have an EID assigned to it, or because the EID is unknown, the destination  
 1713 EID in the message will typically be set to the special null destination EID value.

1714 The Set Endpoint ID command is also used to provide the Physical Address and EID of the Bus Owner to  
 1715 an Endpoint. An Endpoint that needs to communicate with the Bus Owner may capture the physical  
 1716 address and EID that was used to deliver the Set Endpoint ID message.

1717 Note: Endpoints that are not the Bus Owner should not issue the Set Endpoint ID command because it can  
 1718 cause the receiver of the message to capture incorrect information for the Bus Owner's address.

1719 An MCTP bridge may elect to have a single EID for its functionality, rather than using an EID for each port  
 1720 (bus connection) that is connected to a different bus owner. See 8.19.2 for more information. In this case,  
 1721 the bridge will accept its EID assignment from the "first" bus to deliver the Set Endpoint ID request to the  
 1722 bridge.

1723 It is recognized that different internal processing delays within a bridge can cause the temporal ordering  
 1724 of requests to be switched if overlapping requests are received over more than one bus. Therefore, which  
 1725 request is accepted by an implementation is not necessarily tied to the request that is first received at the  
 1726 bridge, but instead will be based on which request is the first to be processed by the bridge.

1727 If an EID has already been assigned and the Set Endpoint ID command is issued from a different bus  
 1728 without forcing an EID assignment, the command shall return a `SUCCESSFUL` completion code, but the  
 1729 response parameters shall return an EID assignment status of "EID rejected".

1730 The Set Endpoint ID command functions in the same manner regardless of whether the endpoint uses a  
 1731 static EID. The only difference is that if an endpoint has a static EID, it uses that EID as its initial "default"  
 1732 EID value. The endpoint does not treat this initial EID as if it were assigned to it by a different bus owner.  
 1733 That is, the endpoint shall accept the EID assignment from the first bus that the command is received  
 1734 from, and shall track that bus as the originating bus for the EID for subsequent instances of Set Endpoint  
 1735 ID command. See 8.17.2 for more information. The request and response parameters are specified in  
 1736 Table 14.

1737

**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</p>



	Byte	Description
		support a Discovered flag.
	2	Endpoint ID.  0xFF, 0x00 = illegal. 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.
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 10.10 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.

1738 **10.4 Get Endpoint ID**

1739 The Get Endpoint ID command returns the EID for an endpoint. This command is typically issued only by  
 1740 a bus owner to retrieve the EID that was assigned to a particular physical address. Thus, the destination  
 1741 EID in the message will typically be set to the special Physical Addressing Only EID value. The request  
 1742 and response parameters are specified in Table 15.

1743 **Table 15 – Get Endpoint ID Message**

	Byte	Description
Request data	–	–
Response data	1	Completion Code.

	Byte	Description
	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 11b = reserved [2:0] = reserved [1:0] = Endpoint ID Type: 00b = dynamic EID. The endpoint uses a dynamic EID only. 01b = static EID supported. 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. The following two status return values are optional. If provided, they must 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. 10b = static EID supported. Present EID matches static EID. The endpoint has been configured with a static EID. The present value is the same as the static value. 11b = static EID supported. Present EID does not match static EID. Endpoint has been configured with a static EID. The present value is different than the static value. See 8.17.2 for more information.
	4	Medium-Specific Information. 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. This byte shall be returned as 0x00 unless otherwise specified by the transport binding.

## 1744 10.5 Get Endpoint UUID Command

1745 The Get Endpoint UUID command returns a universally unique identifier (UUID), also referred to as a  
 1746 globally unique ID (GUID), for the management controller or management device. The command can be  
 1747 used to correlate a device with one or more EIDs. The format of the ID follows the byte (octet) format  
 1748 specified in [RFC4122](#). [RFC4122](#) specifies four different versions of UUID formats and generation  
 1749 algorithms suitable for use for a device UUID in IPMI. These are version 1 (0001b) "time based", and  
 1750 three "name-based" versions: version 3 (0011b) "MD5 hash", version 4 (0100b) "Pseudo-random", and

1751 version 5 "SHA1 hash". The version 1 format is recommended. However, versions 3, 4, or 5 formats are  
 1752 also allowed. A device UUID should never change over the lifetime of the device. The request and  
 1753 response parameters are specified in Table 16.

1754 **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)

1755 The individual fields within the UUID are stored most-significant byte (MSB) first per the convention  
 1756 described in [RFC4122](#). See Table 17 for an example format.

1757 **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	

1758 **10.6 Get MCTP Version Support**

1759 This command can be used to retrieve the MCTP base specification versions that the endpoint supports,  
 1760 and also the message type specification versions supported for each message type. The format of the  
 1761 request and response parameters for this message is given in Table 18.

1762 More than one version number can be returned for a given message type by the Get MCTP Version  
 1763 Support command. This enables the command to be used for reporting different levels of compatibility  
 1764 and backward compatibility with different specification versions. The individual specifications for the given  
 1765 message type define the requirements for which versions number values should be used for that  
 1766 message type. Those documents define which earlier version numbers, if any, must also be listed.

1767 The command returns a completion code that indicates whether the message type number passed in the  
 1768 request is supported or not. This enables the command to also be used to query the endpoint for whether  
 1769 it supports a given message type.

1770 NOTE: Version numbers are listed from oldest to newest. Versioning commands and version formats for vendor-  
 1771 defined message types, 0x7E and 0x7F, are vendor-specific and considered outside the scope of this specification.

1772

1773

**Table 18 – Get MCTP Version Support Message**

	Byte	Description
Request data	1	<p>Message Type Number</p> <p>The Message Type Number to retrieve version information for:</p> <p>0xFF = return MCTP base specification version information.</p> <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>other = return version information for a given message type. See <a href="#">MCTP ID</a> for message type numbers.</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 <a href="#">DSP4004</a> 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>

	Byte	Description
		The encoding of the version number and alpha fields is provided in 10.6.1.
	(7:X)	Version Number Entries 2 through N. Additional 32-bit major/minor version numbers, if any.

### 1774 10.6.1 Version Field Encoding

1775 The version field is comprised of four bytes referred to as the "major", "minor", "update", and "alpha"  
1776 bytes. These bytes shall be encoded as follows:

1777 The "major", "minor", and "update" bytes are BCD-encoded, and each byte holds two BCD digits. The  
1778 "alpha" byte holds an optional alphanumeric character extension that is encoded using one of the  
1779 alphabetic characters [a-z, A-Z] from the US-ASCII ([RFC20](#)) Character Set. The semantics of these fields  
1780 follows that specified in [DSP4004](#).

1781 The value 0x00 in the alpha field means that the alpha field is not used. Software or utilities that display  
1782 the version number should not display any characters for this field.

1783 The value 0xF in the most-significant nibble of a BCD-encoded value indicates that the most-significant  
1784 nibble should be ignored and the overall field treated as a single-digit value. Software or utilities that  
1785 display the number should only display a single digit and should not put in a leading "0" when displaying  
1786 the number.

1787 A value of 0xFF in the "update" field indicates that the field to be ignored. Software or utilities that display  
1788 the version number should not display any characters for the field. 0xFF is not allowed as a value for the  
1789 "major" or "minor" fields.

1790 EXAMPLES:

1791 Version 1.1.0 → 0xF1F1F000

1792 Version 3.1 → 0xF3F1FF00

1793 Version 1.0a → 0xF1F0FF61

1794 Version 3.7.10a → 0xF3F71061

1795 Version 10.11.7 → 0x1011F700

### 1796 10.6.2 MCTP Base Specification Version Number

1797 MCTP implementations that follow this particular specification shall return the following version information in the  
1798 response to the Get MCTP Version Support message when the Message Type parameter in the request is set to  
1799 0xFF (return MCTP base specification version information).

1800 The Version Number Entry 1 field shall be used to indicate backward compatibility with Version 1.0 of the  
1801 base specification as:

1802 **1.0** [Major version 1, minor version 0, no update version, no alpha]

1803 This is reported using the encoding as: 0xF1F0FF00

1804 The Version Number Entry 2 field shall be used to indicate backward compatibility with Version 1.1 of the  
1805 base specification as:

1806 **1.1.0** [Major version 1, minor version 1, update version 0, no alpha]

1807 This is reported using the encoding as: 0xF1F1F000

1808 The version of the MCTP base specification for this specification shall be reported in Version Number  
1809 Entry 3 as:

1810       **1.2.0**   [Major version 1, minor version 2, update version 0, no alpha]

1811 This is reported using the encoding as: 0xF1F2F000

1812

### 1813 **10.6.3 MCTP Control Protocol Version Information**

1814 MCTP implementations that follow this particular specification shall return the following version information in the  
1815 response to the Get MCTP Version Support message when the Message Type parameter in the request is set to  
1816 0x00 (return MCTP control protocol version information).

1817 The Version Number Entry 1 field shall be used to indicate backward compatibility with Version 1.0 of the  
1818 base specification Control Protocol as:

1819       **1.0** [Major version 1, minor version 0, no update version, no alpha]

1820 This is reported using the encoding as: 0xF1F0FF00

1821 The Version Number Entry 2 field shall be used to indicate backward compatibility with Version 1.1 of the  
1822 base specification Control Protocol as:

1823       **1.1.0** [Major version 1, minor version 1, update version 0, no alpha]

1824 This is reported using the encoding as: 0xF1F1F000

1825 The version of the MCTP base specification Control Protocol for this specification shall be reported in  
1826 Version Number Entry 3 as:

1827       **1.2.0**   [Major version 1, minor version 2, update version 0, no alpha]

1828 This is reported using the encoding as: 0xF1F2F000

1829

1830 **10.7 Get Message Type Support**

1831 The Get Message Type Support command enables management controllers to discover the MCTP  
 1832 control protocol capabilities supported by other MCTP endpoints, and get a list of the MCTP message  
 1833 types that are supported by the endpoint. The request and response parameters for this message are  
 1834 listed in Table 19.

1835 The response to this command may be specific according to which bus the request was received over  
 1836 (that is, a device that supports a given message type may not support that message type equally across  
 1837 all buses that connect to the device).

1838 **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 <a href="#">MCTP ID</a> .

1839 **10.8 Get Vendor Defined Message Support**

1840 The Get Vendor Defined Message Support operation enables management controllers to discover  
 1841 whether the endpoint supports vendor-defined messages, and, if so, the vendors or organizations that  
 1842 defined those messages. The format and definition of the request and response parameters for this  
 1843 message is given in Table 20.  
 1844

1845

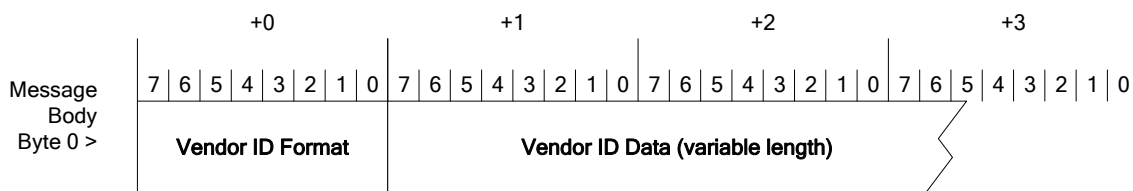
**Table 20 – Get Vendor Defined Message Support Message**

	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 20 – 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.

1846 **10.8.1 Vendor ID Formats**

1847 Figure 20 shows the general structure of Vendor ID fields used in this specification. The first byte of the  
 1848 field contains the Vendor ID Format, a numeric value that indicates the definition space and format of the  
 1849 ID. The remainder of the field holds the Vendor ID Data with content and format as specified in Table 21.

1850 The MCTP management controller or management device can pick which format is best suited for the  
 1851 device. In general, if the device does not already have an existing vendor ID that matches one of the  
 1852 specified formats, it is recommended that the IANA enterprise number format be used.



1853

1854 **Figure 20 – Structure of Vendor ID Field for Get Vendor Defined Capabilities Message**



1855

Table 21 – Vendor ID Formats

Vendor ID Format Name	Vendor ID Format	Vendor ID Data Length	Description
PCI Vendor ID	0x00	2	<b>16-bit Unsigned Integer.</b> 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	<b>32-bit Unsigned Integer.</b> 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.

1856 **10.9 Resolve Endpoint ID**

1857 This command is sent to the bus owner to resolve an EID into the physical address that must be used to  
 1858 deliver MCTP messages to the target endpoint. The command takes an EID as an input parameter in the  
 1859 request and returns the EID and the physical address for routing to that EID (if any) in the response. The  
 1860 response data will also indicate if no mapping was available.

1861 An endpoint knows the physical address of the bus owner by keeping track of which physical address  
 1862 was used when the endpoint received its EID assignment through the Set Endpoint ID command. The  
 1863 endpoint can send this command to the bus owner using the null destination EID value. This eliminates  
 1864 the need for the endpoint to also keep track of the EID of the bus owner. The request and response  
 1865 parameters are specified in Table 22.

1866

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	<p>Physical Address.</p> <p>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.</p>
--	-----	--

## 1867 10.10 Allocate Endpoint IDs

1868 Bus owners are responsible for allocating pools of EIDs to MCTP bridges that are lower in the bus  
 1869 hierarchy. This is done using the Allocate Endpoint IDs command. The EID for the bridge itself is  
 1870 assigned separately and is *not* part of the pool given with this command.

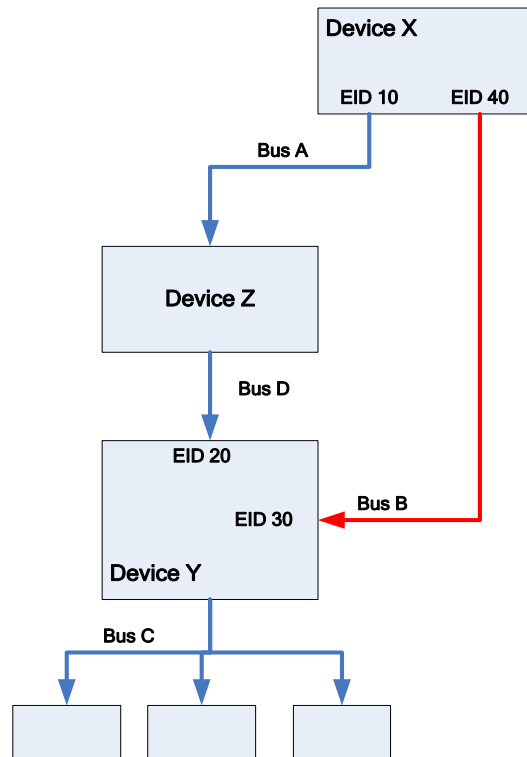
1871 The bus owner will typically use this command as part of the EID assignment process for a bus. When a  
 1872 device has been assigned an EID using the Set Endpoint ID command, the response to that command  
 1873 indicates whether the endpoint supports an EID pool. If the device indicates that it supports an EID pool,  
 1874 the bus owner can then issue the Allocate Endpoint IDs command to supply the pool of EIDs to the  
 1875 device.

1876 NOTE: The Allocate Endpoint IDs command can also cause a bridge to rebuild its routing table. See 10.11.2 for  
 1877 more information.

1878 When an EID or EID pool that was previously allocated becomes unused (for example, due to a hot-swap  
 1879 removal), the bus owner must reclaim the endpoint's EID or EID pool allocation. See 8.17 for additional  
 1880 details.

1881 Referring to Figure 21, there is a potential race condition with handling EID allocation. In the scenario  
 1882 shown in this figure, it is possible that device X and device Z might both be assigning EIDs to device Y at  
 1883 the same time. This also means that, unless steps are taken, device Z could allocate endpoints to device  
 1884 Y only to have this overwritten by a set of endpoints assigned by device X.

1885 To prevent this, the Allocate Endpoint IDs command is only accepted from the "first" bus that provides the  
 1886 EID pool to the device. If another bus owner attempts to deliver an EID pool through another bus, the  
 1887 request will be rejected unless an intentional over-ride is done.



1888

1889

Figure 21 – EID Pools from Multiple Bus Owners

1890 The Allocate Endpoint IDs message fields are described in Table 23.

1891

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 sections.) 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 10.11.2 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)

	Byte	Description
		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 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 section.) 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.

## 1892 10.11 Routing Information Update

1893 The Routing Information Update message is used by a bus owner to give routing information to a bridge  
1894 for the bus on which the message is being received.

1895 Because the physical address format is based on the bus over which the request is delivered, the bus  
1896 owner must use the medium-specific physical address format for the addresses sent using this command.

1897 An MCTP bridge may be sent more than one instance of this command to transfer the update information.  
1898 An integral number of routing information update entries must be provided in the command (that is,  
1899 routing information update entries cannot be split across instances of the command).

1900 **10.11.1 Adding and Replacing Entries**

1901 The recipient of this command must check to see whether the information in the request corresponds to  
 1902 the EID for an existing entry for the bus over which the command was received. If so, it must replace that  
 1903 entry with the new information. If an entry for a given EID or EID range does not already exist, it must  
 1904 create new entries for the given EIDs. In some cases this may require the bridge to split existing entries  
 1905 into multiple entries.

1906 NOTE: A bus owner is only allowed to update entries that correspond to its bus. For each routing table entry that  
 1907 was created or updated through the Routing Information Update message, the bridge must keep track of which bus it  
 1908 received the Routing Information Update from. This is necessary so that when a Routing Information Update is  
 1909 received from a particular bus, the bridge only updates entries that correspond to entries that were originally given to  
 1910 it from that bus.

1911 **10.11.2 Rebuilding Routing Tables**

1912 A bridge that receives and accepts the Allocate Endpoint IDs command with the "Force Allocation" bit set  
 1913 (1b) must clear out and rebuild its routing table information. The bridge shall issue commands to reassign  
 1914 EIDs and re-allocate EID pools to all downstream devices. The request and response parameters are  
 1915 specified in Table 24, and format information is provided in Table 25.

1916 **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

1917 **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

Byte	Description
	data is being provided for.

## 1918 10.12 Get Routing Table Entries

1919 This command can be used to request an MCTP bridge to return data corresponding to its present routing  
 1920 table entries. This data is used to enable troubleshooting the configuration of routing tables and to enable  
 1921 software to draw a logical picture of the MCTP network. More than one instance of this command will  
 1922 typically need to be issued to transfer the entire routing table content.

1923 An integral number of routing table entries must be provided in the response to this command (that is,  
 1924 routing table entries cannot be split across instances of the command). The request and response  
 1925 parameters are specified in Table 26, and format information is provided in Table 27.

1926 **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.

1927 **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 must be reported as separate entries. Dynamically created entries include entries that were generated from the Routing Information Update command as well as entries that were created as a result of the bridge doing EID

Byte	Description
	<p>assignment and EID pool allocation as a bus owner.</p> <p>0b = Entry was dynamically created</p> <p>1b = Entry was statically configured</p> <p>[4:0] – Port number</p> <p>This value is chosen by the bridge device vendor and is used to identify a particular bus connection that the physical address for the entry is defined under. In some cases, this number may correspond to an internal "logical" bus that is not directly connected to an external physical bus. Port numbers are required to be static.</p> <p>It is recommended, but not required, that the ports (bus connections) on the bridge be numbered sequentially starting from 0x00. This specification does not define any requirements or recommendations on how port numbers are assigned to corresponding physical connections on a device.</p>
4	Physical Transport Binding Type 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.

### 1928 10.13 Prepare for Endpoint Discovery

1929 The Endpoint Discovery message is used to determine if devices on a bus communicate MCTP (see  
 1930 Table 28). Whether this message is required depends on the particular medium. Currently, this message  
 1931 may be required only by a particular transport binding, such as PCI Express (PCIe) VDM, because other  
 1932 bindings such as SMBus may use other mechanisms for determining this information.

1933 Each endpoint (except the bus owner) on the bus maintains an internal flag called the "Discovered" flag.

1934 The Prepare for Endpoint Discovery command is issued as a broadcast Request message on a given bus  
 1935 that causes each endpoint on the bus to set their respective Discovered flag to the "undiscovered" state.  
 1936 The flag is subsequently set to the "discovered" state when the Set Endpoint ID command is received by  
 1937 the endpoint.

1938 An endpoint also sets the flag to the "undiscovered" state at the following times:

- 1939 • Whenever the physical address associated with the endpoint changes or is assigned
- 1940 • Whenever an endpoint first appears on the bus and requires an EID assignment
- 1941 • During operation if an endpoint enters a state that requires its EID to be reassigned
- 1942 • For hot-plug endpoints: After exiting any temporary state where the hot-plug endpoint was  
 1943 unable to respond to MCTP control requests for more than  $T_{RECLAIM}$  seconds (where  $T_{RECLAIM}$  is  
 1944 specified in the physical transport binding specification for the medium used to access the  
 1945 endpoint). See 8.17.5 for additional information.

1946 Only endpoints that have their Discovered flag set to "undiscovered" will respond to the Endpoint  
 1947 Discovery message. Endpoints that have the flag set to "discovered" will not respond.

1948 The destination EID for the Prepare for Endpoint Discovery message is set to the Broadcast EID value  
 1949 (see Table 2) in the request message to indicate that this is a broadcast message. The response  
 1950 message sets the destination EID to be the ID of the source of the request message, which is typically the  
 1951 EID of the bus owner. The request and response parameters are specified in Table 28.

1952 The Prepare for Endpoint Discovery message has no effect on existing EID assignments. That is,  
 1953 endpoints shall normally retain their EIDs until they are explicitly changed via the Set Endpoint ID  
 1954 command, and shall not clear them after getting a "Prepare for Endpoint Discovery" command. (Note that  
 1955 endpoints may lose their EIDs under other conditions such as power state changes, etc., as described  
 1956 elsewhere in this specification.)

1957 The Endpoint Discovery and Prepare for Endpoint Discovery commands may only be supported on  
 1958 particular transport bindings (e.g. PCIe). If the binding does not use this discovery approach (e.g. SMBus)  
 1959 the endpoint shall return an `ERROR_UNSUPPORTED_CMD` completion status for those commands.

1960 **Table 28 – Prepare for Endpoint Discovery Message**

	Byte	Description
Request data	–	–
Response data	1	Completion Code

## 1961 **10.14 Endpoint Discovery**

1962 This command is used to discover endpoints that have their Discovered flag set to "undiscovered". Only  
 1963 endpoints that have their Discovered flag set to "undiscovered" will respond to this message. Endpoints  
 1964 that have the flag set to "discovered" will not respond.

1965 This message is typically sent as a Broadcast Request message by the bus owner using the Broadcast  
 1966 EID as the destination EID, though for testing purposes endpoints must also accept and handle this  
 1967 command as a non-broadcast Request. Additionally, the request may be sent as a datagram, depending  
 1968 on the transport binding requirements. The request and response (if any) parameters are specified in  
 1969 Table 29.

1970 **Table 29 – Endpoint Discovery Message**

	Byte	Description
Request data	–	–
Response data	1	Completion Code

## 1971 **10.15 Discovery Notify**

1972 This message is available for use as a common message for enabling an endpoint to announce its  
 1973 presence to the bus owner. This will typically be used as part of the endpoint discovery process when an  
 1974 MCTP device is hot-plugged onto or becomes powered up on an MCTP bus.

1975 Whether and how this message is used for endpoint discovery depends on the particular physical  
 1976 transport binding specification. For example, the SMBus transport binding does not use this message for  
 1977 an endpoint to announce itself because it takes advantage of mechanisms that are already defined for  
 1978 SMBus.

1979 This message should only be sent from endpoints to the bus owner for the bus that the endpoint is on so  
 1980 it can notify the bus owner that the endpoint has come online and may require an EID assignment or



1981 update. Additionally, the request may be sent as a datagram, depending on the transport binding  
 1982 requirements. The request and response (if any) parameters are specified in Table 30.

1983 **Table 30 – Discovery Notify Message**

	Byte	Description
Request data	–	–
Response data	1	Completion Code

## 1984 10.16 Get Network ID Command

1985 The Get Network ID command returns a universally unique identifier (UUID), also referred to as a globally  
 1986 unique ID (GUID), for a given MCTP network. Typically this command is sent to the topmost MCTP bus-  
 1987 owner since the topmost bus-owner has this knowledge. A Network ID is required for add-in MCTP  
 1988 networks (For example, an MCTP Network on an add-in card or module). A Network ID is not required for  
 1989 a fixed (not add-in) MCTP network provided there is only one network in the system implementation. A  
 1990 Network ID is required for fixed MCTP networks when more than one fixed network exists in the system  
 1991 implementation and is simultaneously accessible by a common entity such as system software.

1992 The format of the ID follows the byte (octet) format specified in [RFC4122](#). [RFC4122](#) specifies four  
 1993 different versions of UUID formats and generation algorithms suitable for use for a device UUID in IPMI.  
 1994 These are version 1 (0001b) "time based", and three "name-based" versions: version 3 (0011b) "MD5  
 1995 hash", version 4 (0100b) "Pseudo-random", and version 5 "SHA1 hash". The version 1 format is  
 1996 recommended. However, versions 3, 4, or 5 formats are also allowed. A device UUID should never  
 1997 change over the lifetime of the device. The request and response parameters are specified in Table 16.

1998 **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)

1999 The individual fields within the UUID are stored most-significant byte (MSB) first per the convention  
 2000 described in [RFC4122](#). See Table 17 for an example format.

2001

2002 **10.17 Query Hop**

2003 This command can be used to query a bridge to find out whether a given EID must be accessed by going  
 2004 through that bridge, and if so, whether yet another bridge must be passed through in the path to the  
 2005 endpoint, or if the endpoint is on a bus that is directly connected to the bridge.

2006 The command also returns the information about the transmission unit information that the bridge  
 2007 supports in routing to the given target endpoint from the bus that the request was received over. See  
 2008 section 8.23 for more information.

2009 NOTE: The physical transport binding for MCTP may place additional requirements on the physical packet sizes  
 2010 that can be used to transfer MCTP packet payloads, such as requiring that physical packet sizes be in 32-byte or 64-  
 2011 byte increments, or particular power of 2 increments (for example, 128, 256, 512, and so on).

2012 The request and response parameters are specified in Table 32.

2013

**Table 32 – Query Hop Message**

	Byte	Description
Request data	1	Target Endpoint ID  0x00, 0xFF = reserved. (An ERROR_INVALID_DATA 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.
Response data	1	Completion Code  An ERROR_INVALID_DATA completion code must 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 must 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 0xFF 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 (0x0000 = 64 bytes, 0x0001 = 80 bytes, and so on).

	Byte	Description
	5:6	Maximum supported outgoing transmission unit size in increments of 16 bytes, starting from the baseline transmission unit (0x0000 = 64 bytes, 0x0001 = 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.

2014 **10.18 Resolve UUID**

2015 This command is used to get information about an endpoint based on its UUID. This command may be  
 2016 sent from any endpoint to the bus owner. This command takes a UUID as a parameter in the request and  
 2017 returns a list of EIDs and physical addresses that matches this UUID.

2018 A bus owner that supports this command shall keep in the routing table entries the UUID of each of the  
 2019 endpoints. The UUID values can be found using a “Get Endpoint UUID” command.

2020 An endpoint knows the physical address of the bus owner by keeping track of which physical address  
 2021 was used when the endpoint received its EID assignment through the Set Endpoint ID command. The  
 2022 endpoint can send this command to the bus owner using the null destination EID value. This eliminates  
 2023 the need for the endpoint to also keep track of the EID of the bus owner. The request and response  
 2024 parameters are specified in Table 12.

2025 **Table 12 – Resolve UUID Message**

	Byte	Description
Request data	1:16	Requested UUID
	17	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 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 13. This field will be absent if the number of EID entries is 0x00.

2026 **Table 13 – Resolve UUID Message Entry Format**

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

2027

2028 **10.19 Transport Specific**

2029 Transport Specific commands are a range of commands that are available for use by transport binding  
 2030 specifications in order to perform additional MCTP Control functions that are defined by a particular  
 2031 transport binding. Transport specific commands shall only be addressed to endpoints on the same  
 2032 medium. A bridge is allowed to block transport specific commands from being bridged to different media.

2033 The request and response parameters are specified in Table 33.

2034 **Table 33 – 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 <a href="#">MCTP ID</a> 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 <a href="#">MCTP ID</a> 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 10.8.1 for the information on the IANA Enterprise ID as used in this specification.
Response data	1	Completion Code

2035 **11 Vendor Defined – PCI and Vendor Defined – IANA Messages**

2036 The Vendor Defined – PCI and Vendor Defined – IANA message types provide a mechanism for  
 2037 providing an MCTP message namespace for vendor-specific messages over MCTP.

2038 The PCI and IANA designations refer to the mechanism that is used to identify the vendor or organization  
 2039 this is specifying the message’s functionality and any parametric data or other fields provided in the  
 2040 message body.

2041 Note that this specification only defines the initial bytes in the message body of these messages, and sets  
 2042 the requirement that these messages must follow the requirements set by the MCTP base protocol and  
 2043 any additional requirements necessary to meet the transport of these messages over a particular  
 2044 medium, such as path transmission unit limitations.

2045 Otherwise, any other field definitions and higher level message behavior such as retries, error/completion  
 2046 codes, and so on, is message type-specific and thus is vendor-specific.

2047

2048 **11.1 Vendor Defined – PCI Message Format**

2049 For these messages, the MCTP message type is set to the value for "Vendor Defined – PCI" as defined in  
 2050 Table 3. The request and response parameters are specified in Table 34.

2051 **Table 34 – Vendor Defined – PCI Message Format**

	Byte	Description
Request data	1:2	PCI/PCIe Vendor ID. Refer to <a href="#">PCIe</a> . MSB first. This value is formatted per the Vendor Data Field for the PCI Express vendor ID format. See 10.8.1". 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 <a href="#">PCIe</a> . MSB first.
	(3:M)	Vendor-Defined Message Body. 0 to M bytes.

2052 **11.2 Vendor Defined – IANA Message Format**

2053 For these messages, the MCTP message type is set to the value for "Vendor Defined – IANA" as defined  
 2054 in Table 3. The request and response parameters are specified in Table 35.

2055 **Table 35 – 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 10.8.1. 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.

2056

## ANNEX A (informative)

### Notation

2057  
2058  
2059  
2060

#### 2061 A.1 Notations

2062 Examples of notations used in this document are as follows:

- 2063 • 2:N In field descriptions, this will typically be used to represent a range of byte offsets  
2064 starting from byte two and continuing to and including byte N. The lowest offset is on  
2065 the left, and the highest is on the right.
- 2066 • (6) Parentheses around a single number can be used in message field descriptions to  
2067 indicate a byte field that may be present or absent.
- 2068 • (3:6) Parentheses around a field consisting of a range of bytes indicates the entire range  
2069 may be present or absent. The lowest offset is on the left, and the highest is on the  
2070 right.
- 2071 • [PCle](#) Underlined, blue text is typically used to indicate a reference to a document or  
2072 specification called out in 2, "Normative References", or to items hyperlinked within  
2073 the document.
- 2074 • rsvd Abbreviation for Reserved. Case insensitive.
- 2075 • [4] Square brackets around a number are typically used to indicate a bit offset. Bit offsets  
2076 are given as zero-based values (that is, the least significant bit [LSb] offset = 0).
- 2077 • [7:5] A range of bit offsets. The most-significant is on the left, and the least-significant is on  
2078 the right.
- 2079 • 1b The lower case "b" following a number consisting of 0s and 1s is used to indicate the  
2080 number is being given in binary format.
- 2081 • 0x12A A leading "0x" is used to indicate a number given in hexadecimal format.

2082

## ANNEX B (informative)

### Change Log

Version	Date	Author	Description
1.0.0	05/21/2009	T. Slaight	
1.1.0	2/19/2010	T. Slaight	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	01/24/2013	T. Slaight	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.

2083  
2084  
2085  
2086  
2087

2088