UNH–IOL
NVMe Testing Service

Test Plan for NVMe-MI Conformance
Version 12.0
Target Specification: NVMe-MI 1.0a
Technical Document

NOTICE: This is a living document. All contents are subject to change.
Individual tests and/or test groups may be added/deleted/renumbered in forthcoming revisions.
General feedback and comments are welcome through the NVMe Consortium at UNH–IOL.

Last Updated: April 29, 2019

UNH–IOL NVMe Testing Service
21 Madbury Rd Suite 100
Durham, NH 03824

Tel: +1 603–862–0090
Fax: +1 603–862–4181
Email: nvmelab@iol.unh.edu

TABLE OF CONTENTS

MODIFICATION RECORD

2017 February 7 (Version 7.0 r01) Initial Release
David Woolf:

2017 February 14 (Version 7.0 r02) Initial Release
David Woolf: Adjusted Group 1, 2, 3 to be FYI.

2017 February 21 (Version 7.0 r03) Initial Release
David Woolf: Adjusted Group 9, 10, 11, 12 to be In Progress.

March 22, 2017 (Version 7.0)
David Woolf: Final version published to UNH-IOL site ahead of May 2017 NVMe Plugfest #7.

April 4, 2017 (Version 7.0a)
David Woolf: Updated tests 4.7, 4.10, 5.3, 7.3 to be FYI for May 2017 NVMe Plugfest #7 per direction of NVMe-MI WG.

August 29, 2017 (Version 8.0)
David Woolf: Merged original Test Specification v0.9 from NVMe-MI Working Group into this document. Made Test 8.5 FYI.

September 12, 2017 (Version 8.0a)
David Woolf: Added Appendix D on Test Tool versions supporting the tests described in this test document.

September 19, 2017 (Version 9.0 draft)
David Woolf:

- Edited procedure and observable results for Test 8.5 to accommodate new requirements published in NVMe-MI 1.0 ECN 003.

February 1, 2018 (Version 9.0 draft)
David Woolf:

- Edited observable results for Test 4.8 to accommodate for varied error responses allowed by chapter 4.2 of the NVMe-MI 1.0 specification.
- Updated tests 4.10 and 5.3 from FYI to Mandatory.
- Modified Test 8.5 to include separate cases for devices which have or have not implemented NVMe-MI 1.0 ECN 003.

August 30, 2018 (Version 10.0 release)
David Woolf:

- Release for 10.0.

November 26, 2018 (Version 11.0 release)
David Woolf:

1. Updated Test 8.5 Case 1 and Test 8.5 Case 2 to accommodate NVMe-MI 1.0 ECN 003 changing the RENT field from being a zeroes based value to not being a zeroes based value. Products implementing NVMe-MI 1.0 may or may not implement NVMe-MI 1.0 ECN 003. Products implementing NVMe-MI 1.0a or higher must implement NVME-MI 1.0 ECN 003.
2. Fixed formatting for sub test cases in Test 8.7.

April 29, 2019 (Version 12.0 release)
David Woolf:

- Updated Test 7.3 to reflect proper use of the CESF bit, per TP2008.
- Updated section on NVMe Integrators List requirements to match the NVMe Integrators List Policy Document.

ACKNOWLEDGMENTS

The UNH–IOL would like to acknowledge the efforts of the following individuals in the development of this test plan:

Gordon Getty	Teledyne-LeCroy

INTRODUCTION

The University of New Hampshire's InterOperability Laboratory (IOL) is an institution designed to improve the interoperability of standards–based products by providing a neutral environment where a product can be tested against other implementations of a common standard, both in terms of interoperability and conformance. This particular suite of tests has been developed to help implementers evaluate the NVMe-MI functionality of their products. This test suite is aimed at validating products in support of the work being directed by the NVMe Organization.

These tests are designed to determine if a product conforms to specifications defined in the NVMe Management Interface specification, hereafter referred to as the "NVMe-MI Specification"). Successful completion of these tests provide a reasonable level of confidence that the Device Under Test (DUT) will function properly in many NVMe-MI environments.

REFERENCES

The following documents are referenced in this text:

1. NVMe Management Interface 1.0a Specification, April 8 2017

Group 1: MCTP Base Tests

Test 1.1 – MCTP Endpoint ID – (FYI)

Purpose: Source Endpoint ID should be set to the assigned value

References:
MCTP Base Specification Section 8.2

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get Endpoint ID MCTP command
3. Wait for response.

Observable Results:
1. Ensure that the Endpoint ID returned by Get Endpoint ID command is the one assigned during MCTP initialization. f returned correctly, then result is PASS, otherwise, result is FAIL

Possible Problems: None

Test 1.1.1 – MCTP Endpoint ID Response – Reserved Bits – (FYI)

Purpose: The intent of this test is to ensure that all the reserved bits of the response are set to zero in the Get Endpoint ID Response

References:
MCTP Base Specification Section 8.2

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get EndPointID command
3. Wait for the Response Message

Observable Results:
1. Ensure that the Get Endpoint ID command returns with successful completion.
2. Ensure that the following bits are reserved:
    a. Bits 7:6 of byte 3 are reserved.
    b. Bits 2:0 of byte 3 are reserved.
    c. Endpoint Type (Bits 5:4 of byte 3) have values 10b and 11b reserved.

Possible Problems: None

Test 1.2 – MCTP Packet Sequence Number – (FYI)

Purpose: 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.

References:
MCTP Base Specification Section 8.2

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get Transmission Unit Size
3. Wait for Response Message
4. Execute NVMe-MI VPD Read with Data Offset = 0 and Data Length = 256.
5. Wait for Response

Observable Results:
1. Ensure that the Get Endpoint ID command returns with successful completion.
2. Ensure that the Get Transmission Unit Size returns the default value of 64
3. Ensure that the response of VPD Read is split into 5 MCTP packets.
4. Ensure that the sequence number of the packets are a modulo of 4 and are in order.

Possible Problems: None

Test 1.3 – MCTP Tag Owner and Message Tag Bits – (FYI)

Purpose: 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.

References:
MCTP Base Specification Section 8.2

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get Transmission Unit Size
3. Wait for Response Message
4. Execute NVMe-MI VPD Read with Data Offset = 0 and Data Length = 256.
5. Wait for Response;

Observable Results:
1. Ensure that the Get Endpoint ID command returns with successful completion.
2. Ensure that the Get Transmission Unit Size returns the default value of 64
3. Ensure that the Tag Owner and Message Bits of the VPD Read response MCTP packets are the same,
1. Starting from the First message (SOM = 1, EOM = 0) to the last packet (SOM = 0, EOM = 1).

Possible Problems: None

Test 1.4.1 – MCTP Bad Packet 1 – (FYI)

Purpose: Individual packets are dropped (silently discarded) by an endpoint under the following conditions. These packets are discarded before being checked for acceptance or rejection for message assembly. Therefore, these packets will not cause a message assembly to be started or terminated.
Test case 1.4.1: Unexpected “middle” packet or “end” packet will be properly handled

References:
MCTP Base Specification Section 8.6

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get Endpoint ID command, wait for response
3. Send MCTP packet (Get Endpoint ID command) with SOM and EOM not set ("middle" packet)
4. Wait for 126 milliseconds (MT2, response timeout)
5. Execute Get Endpoint ID command, wait for response
6. Send MCTP packet (Get Endpoint ID command) with SOM not set and EOM set ("end" packet)
7. Wait for 126 milliseconds (MT2, response timeout)
8. Execute Get Endpoint ID command, wait for response

Observable Results:
1. Ensure that the Get Endpoint ID commands with "middle" packet designation is silently dropped by the DUT.
2. Ensure that the Get Endpoint ID commands with "end" packet designation is silently dropped by the DUT.
3. Ensure that the proper responses for correct Get Endpoint ID commands are returned by the DUT.
1. If all of the above true, then result is PASS, otherwise result is FAIL

Possible Problems: None

Test 1.4.2 – MCTP Bad Packet 2 – (FYI)

Purpose:
Bad Message Integrity Check – Ensure that all Get EndPointID requests with a bad Message Integrity Check are dropped silently by the DUT
These packets are discarded before being checked for acceptance or rejection for message assembly.
Therefore, these packets will not cause a message assembly to be started or terminated.

References:
MCTP Base Specification Section 8.8

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Configuration Get command
3. Wait for the Response Message
4. Execute Configuration Get with a wrong Message Integrity Check
5. Execute Configuration Get command
6. Wait for the Response Message

Observable Results:
1. Ensure that the Configuration Get returns a successful completion.
2. Ensure that the Configuration Get with wrong message integrity check is dropped silently.
3. Ensure that the Last Configuration Get returns a successful completion.

Possible Problems: None

Test 1.4.3 – MCTP Bad Packet 3 – (FYI)

Purpose: Individual packets are dropped (silently discarded) by an endpoint under the following conditions. These packets are discarded before being checked for acceptance or rejection for message assembly. Therefore, these packets will not cause a message assembly to be started or terminated.
Test 1.4.3: Bad, unexpected, or expired message tag is handled correctly.

References:
MCTP Base Specification Section 8.6

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get Endpoint ID command, wait for response
3. Send MCTP packet (Get Endpoint ID command) with TO bit set to zero
4. Wait for 126 milliseconds (MT2, response timeout)
5. Execute Get Endpoint ID command, wait for response

Observable Results:
1. Ensure that the Get Endpoint ID command with bad tag owner is silently dropped by the DUT.
2.
3. Ensure that the proper responses for correct Get Endpoint ID commands are returned by the DUT. If all of the above true, then result is PASS, otherwise result is FAIL

Possible Problems: None

Test 1.4.4 – MCTP Bad Packet 4 – (FYI)

Purpose: Individual packets are dropped (silently discarded) by an endpoint under the following conditions. These packets are discarded before being checked for acceptance or rejection for message assembly. Therefore, these packets will not cause a message assembly to be started or terminated.
Test 1.4.4: Unknown Destination EID.

References:
MCTP Base Specification Section 8.6

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get EndPointID command
3. Wait for the Response Message
4. Repeat Get Endpoint Id command with Destination EID set to 0x11 (invalid), 0x22 (invalid), 0x33 (invalid),0xAB (valid)

Observable Results:
1. Ensure that the Get Endpoint ID command returns with successful completion.
2. Ensure that the Get Endpoint IDs with a different Destination EPID is dropped silently by the DUT.
3. Ensure that the last Get Endpoint ID command returns a successful completion

Possible Problems: None

Test 1.4.5 – MCTP Bad Packet 5 – (FYI)

Purpose: Individual packets are dropped (silently discarded) by an endpoint under the following conditions. These packets are discarded before being checked for acceptance or rejection for message assembly. Therefore, these packets will not cause a message assembly to be started or terminated.
Test 1.4.5: Bad header version is handled correctly

References:
MCTP Base Specification Section 8.6

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get Endpoint ID command, wait for response
3. Send MCTP packet (Get Endpoint ID command) with bad header version (set to 2 instead of 1)
4. Wait for 126 milliseconds (MT2, response timeout)
5. Execute Get Endpoint ID command, wait for response

Observable Results:
1. Ensure that the Get Endpoint ID command with bad header version is silently dropped by the DUT.
2. Ensure that the proper responses for correct Get Endpoint ID commands are returned by the DUT.
3. If all of the above true, then result is PASS, otherwise result is FAIL

Possible Problems: None

Group 2: MCTP Control Message Tests

Test 2.1 – MCTP Control Instance ID – (FYI)

Purpose: Instance ID in the MCTP Response message should be set to the same value as in the MCTP Request message

References:
MCTP Base Specification Section 10.3

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get EndPointID command
3. Wait for the Response Message
4. Execute Get EndPointID command with instance ID set to 0x6, 0xC, 0x11, 0x1F

Observable Results:
1. Ensure that the Get Endpoint ID command returns with successful completion.
2. Ensure that the Datagram Bit is set to 0 and that the instance ID of the Get EndPointID Request and the Instance id of the Get EndPointID of the Response are the same.
3. Repeat the same for all instance IDs.

Possible Problems: None

Group 3: MCTP Commands

Test 3.1 – MCTP Command Set Endpoint ID – (FYI)

Purpose: The intention of this test is to ensure that proper Completion Messages are returned for mandatory MCTP Command Control Messages

References:
MCTP Base Specification Section 11.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization

Observable Results:
1. Ensure that the response to MCTP Command Set Endpoint ID returns with successful completion.
2. Ensure that Bits 7:6 of byte 2 are reserved
3. Ensure that Bits 5:4 of byte 2 of response can only have 00b and 01b as values, 10b and 11b are reserved.
4. Ensure that Bits 3:2 of byte 2 are reserved.
5. Ensure that Bits 1:0 of Byte 2 can only have 00b, 10b, and 01b as values, 11b is reserved.
6. Ensure that if Endpoint ID Assignment Status == 00b, EID Setting (byte 3) = 0xAB

Possible Problems: None

Test 3.2 – MCTP Command Get MCTP Version – (FYI)

Purpose: The intention of this test is to ensure that proper Completion Messages are returned for mandatory MCTP Command Control Messages

References:
MCTP Base Specification Section 11.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization.
2. Perform MCTP Get Version Support with Message Type Number set to 0xFF (MCTP base specification Version information).
3. Wait for Response.
4. Perform MCTP Get Version Support with Message Type Number set to 0x00(MCTP Control Protocol Message Version Information).
5. Wait for Response.
6. Perform MCTP Get Version Support with Message Type Number set to 0x10(MCTP Control Protocol Message Version Information).
7. Wait for Response.
8. Perform MCTP Get Version Support with Message Type Number set to 0x70(MCTP Control protocol message version information)
9. Wait for Response.

Observable Results:
1. Ensure that the response to MCTP Command Get MCTP version support returns with successful completion and return Message Type not supported if Completion code is 0x80.
2. Ensure that the number of Version number entries (32 bit version numbers) is equal to the Version Number entry Count (one based).
3. Repeat the above for Message Type Number = 0x00.
4. Ensure that for Message Type Number = 0x10 and 0x70, the Completion Code in Response Data is set to 0x80.

Possible Problems: None

Test 3.3 – MCTP Command Get Message Type – (FYI)

Purpose: The intention of this test is to ensure that proper Completion Messages are returned for mandatory MCTP Command Control Messages

References:
MCTP Base Specification Section 11.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization.
2. Perform MCTP Get Message Type Support.
3. Wait for Response.

Observable Results:
1. Ensure that the response to MCTP Command Get Message Type returns with successful completion. Ensure that the Number of bytes in the Message Data corresponds to the MCTP Message Type Count in the Response Data.i.e.: MCTP Message Type Count = (Length of Response Data) - 1 - 1.

Possible Problems: None

Test 3.4 – MCTP Command Prepare for Endpoint Discovery – (FYI)

Purpose: The intention of this test is to ensure that proper Completion Messages are returned for mandatory MCTP Command Control Messages

References:
MCTP Base Specification Section 11.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization.

Observable Results:
1. Ensure that the response to MCTP Command Prepare for EndPoint Discovery returns with successful completion.

Note: SMBus should return ERROR_UNSUPPORTED_CMD as Completion Code.

Possible Problems: None

Test 3.5 – MCTP Command Endpoint Discovery – (FYI)

Purpose: The intention of this test is to ensure that proper Completion Messages are returned for mandatory MCTP Command Control Messages

References:
MCTP Base Specification Section 11.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization.

Observable Results:
1. Ensure that the response to MCTP Command EndPoint Discovery returns with successful completion.

Possible Problems: None

Test 3.6 – MCTP Command Get Endpoint ID – (FYI)

Purpose: The intention of this test is to ensure that proper Completion Messages are returned for mandatory MCTP Command Control Messages

References:
MCTP Base Specification Section 11.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization

Observable Results:
1. Ensure the response returns with Successful Completion and that all the reserved bits of the response are set to zero in the Get EndPointID Response.

Possible Problems: None

Group 4: NVMe Error Handling

Test 4.1 – NVMe-MI Invalid Opcode – (Mandatory)

Purpose: Invalid Command Opcode Status response should be sent when the request Opcode is set to 08h – BFh

References:
NVMe-MI Base Specification

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Configuration Get MI Command (Cfg ID 0x1) to the Management Endpoint Command Slot 0, wait for response
3. Perform Configuration Get MI Command (Cfg ID 0x1) to the Management Endpoint Command Slot 1, wait for response
4. Perform Management Interface Command with opcode 0x08 to the Management Endpoint Command Slot 0, wait for response
5. Perform Management Interface Command with opcode 0x29 to the Management Endpoint Command Slot 1, wait for response
6. Perform Management Interface Command with opcode 0x5A to the Management Endpoint Command Slot 0, wait for response
7. Perform Management Interface Command with opcode 0xBE to the Management Endpoint Command Slot 1, wait for response

Observable Results:
1. Ensure that for the Configuration Get commands the Response Messages are returned properly.
2. Ensure that for the commands with reserved opcodes the Response Messages are returned with Invalid Command Opcode Status.
3. If returned correctly, then result is PASS, otherwise, result is FAIL

Possible Problems: None

Test 4.2 – NVMe-MI Reserved Identifier – (Mandatory)

Purpose: Configuration Get: Specifying a reserved identifier in the Configuration Identifier field causes the command to complete with an Invalid Parameter error status.

References:
NVMe-MI Specification

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Configuration Get Command requesting MCTP Transmission Unit Size
3. Wait for the Response message
4. Perform Configuration Get Command with Configuration Identifier set to 0
5. Wait for the Response message
6. Repeat the above for Configuration Identifier values of 0x04, 0x56, 0xBF

Observable Results:
1. Ensure that for the first command proper response is returned, and status is set to Success;
2. Ensure that for the rest of the commands:
    a. Invalid Parameter Error Response is sent
    b. In the Parameter Error Location field of the response, the byte position is set to 8 and bit position - to 0.

Possible Problems: None

Test 4.3 – NVMe-MI Health Status Change – (Mandatory)

Purpose: For Health Status Change (Configuration Identifier 02h) - A Management Endpoint shall complete a Configuration Get command on this Configuration Identifier with a Success Response Message. The NVMe Management Response field is reserved and there is no Response Data.

References:
NVMe-MI Specification

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization
2. Execute Configuration Get with Configuration Identifier set to 0x02(Health Status Change
3. Wait for Response message

Observable Results:
1. Ensure that Configuration get command returns with successful completion.
2. Ensure that the NVMe Management Response field is reserved and that no Response data was returned.
3. (NVMe Management Response Field - bits 8:31 of dw1 of the response)

Possible Problems: None

Test 4.4 – NVMe-MI Reserved Configuration Identifier – (Mandatory)

Purpose: Configuration Set:

- Specifying a reserved identifier in the Configuration Identifier field causes the command to complete with an Invalid Parameter error status.- SMBus/I2C Frequency (Configuration Identifier 01h). If the specified frequency is not supported or the Port Identifier specified is not an SMBus/I2C port, the Management Endpoint shall respond with an Invalid Parameter error status.
(Can use Reserved value of 0 or 4-F)
- MCTP Transmission Unit Size (Configuration Identifier 03h). If the specified MCTP Transmission Unit Size is not supported or the Port Identifier specified is not valid, the Management Endpoint shall abort the command and send a Response Message with an Invalid Parameter error status.

References:
NVMe-MI Specification

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization
2. (SMBus only) Perform Configuration Get to ensure SMBus is supported
3. Perform Configuration Set Command with Configuration Identifier 0x03 (MCTP Transmission Unit Size: Size = 64 bytes)
4. Wait for the Response message
5. Perform Configuration Set Command with Configuration Identifier set to 0
6. Wait for the Response message
7. Repeat the above for Configuration Identifier values of 0x04, 0x56, 0xBF
8. Perform Configuration Set Command with Configuration Identifier set to SMBus/I2C Frequency and Frequency Set to 100 kHz
9. Wait for Response Message
10. Repeat the above for Frequency set to 0h, 4h, and Fh and non- SMBus/I2C port

Observable Results:
1. Ensure that the Set Configuration command returns with successful completion.
2. Ensure that for the rest of the commands with invalid Configuration Identifier, Invalid Parameter Error Response is sent.
3. Ensure that Set Configuration - SMBus/I2C Frequency command returns with successful completion if Frequency is supported, otherwise should return Invalid Parameter response
4. Ensure that for the rest of the commands an Invalid Parameter error response is sent.

Possible Problems: None

Test 4.5 – NVMe-MI MAXRENT Error – (Mandatory)

Purpose: Controller Health Status Poll: For Maximum Response Entries (MAXRENT) in Request message. Specifying 256 entries is interpreted as an Invalid Field. Command to complete with an Invalid Parameter error status.

References:
NVMe-MI Specification

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: Feb 3 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Controller Health Status Poll Command with the following parameters:
    a. -Report all set:
    b. Include PCI Functions set
    c. Controller Status Changes set
    d. -MAXRENT set to 0xFF (which results in 256 entries, as this is zero-based value), all other parameters set to zero.
3. Wait for the Response message;

Observable Results:
1. Ensure that the proper response is returned, and status is set to Invalid Parameter;
2. Ensure that for the next command:
    a. Invalid Parameter Response is set
    b. In the Parameter Error Location field of the response the byte position is set to 10 and bit position - to 0.

Possible Problems: None

Test 4.6 – NVMe-MI Reserved Data Structure Type – (Mandatory)

Purpose: Read NVMe-MI Data Structure:

- If Data Structure Type (DTYP) in Request is set to reserved value, the command shall complete with an Invalid Parameter error status. (Implicit)

References:
NVMe-MI Specification

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: Feb 3 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Read NVMe-MI Data Structure Command with Data Structure Type set to 0x05
3. Wait for the Response message
4. Repeat the above Data Structure Type values of 0x23, 0x56, 0x9A, 0xFF.

Observable Results:
1. Ensure that for each command the proper response is returned, and status is set to Invalid Parameter;
2. Ensure that for the rest of the commands:
    a. Invalid Parameter Error Response is sent
    b. In the Parameter Error Location field of the response the byte position is set to 11 and bit position - to 0.

Possible Problems: None

Test 4.7 – NVMe-MI Invalid VPD Read Size -(FYI)

Purpose: VPD Read:

- If the Data Length plus Data Offset fields are greater than the size of the VPD, then the Management Endpoint does not return the VPD contents and responds with an Invalid Parameter error status response

References:
NVMe-MI Specification

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Read VPD with Data Length set to 7 bytes and Offset to 1 byte
3. Wait for the Response message
4. Perform Read VPD with Data Length set to 5 bytes and Offset to 260 bytes
5. Wait for the Response message
6. Perform Read VPD with Data Length set to 300 bytes and Offset to 5 bytes
7. Wait for the Response message

Observable Results:
1. Ensure that for First Command proper response is returned and status is set to 'success'
2. Ensure that for the rest of the commands:Invalid Parameter Error Response is sent;
3. In the Parameter Error Location field of the response the byte position is set to 8 (0x08) and bit position - to 0.
    a. flagging the length field in NVMe management DWord 1 would be correct
    b. If the offset is not within range then the byte position 8 should be returned. If the offset is good, but the length causes the boundary to be exceeded then 0xc should be the byte indication

Possible Problems: None

Test 4.8 – NVMe-MI Invalid VPD Write Status – (Mandatory)

Purpose: VPD Write:
The intent of this test is to ensure that the VPD Write returns Parameter Error Response when the Data length and Data Offset Fields greater than the size of VPD.

References:
NVMe-MI Specification Chapter 4.2

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: January 29, 2018

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Write VPD with Data Length set to 7 bytes and Offset to 1 byte
3. Wait for the Response message
4. Perform Write VPD with Data Length set to 5 bytes and Offset to 260 bytes
5. Wait for the Response message
6. Perform Write VPD with Data Length set to 300 bytes and Offset to 5 bytes
7. Wait for the Response message

Observable Results:
1. Ensure that for First Command proper response is returned and status is set to 'success'.
2. Ensure that for the rest of the commands:
    a. Invalid Parameter Error Response is sent
    b. In the Parameter Error Location field of the response the byte position is set to 10 (0x0A) and bit position - to 0.
    c. If DOFST is out of bounds then 0x8 should be returned If DOFST is valid and DLEN causes too large, then 0xC is returned.
3. If there are multiple error conditions detected by the DUT, then the response may be an error code other than 'Invalid Parameter Error'. This is acceptable behavior.

Possible Problems: None

Test 4.9 – NVMe-MI Invalid Parameter Status – (Mandatory)

Purpose: Invalid Parameter Status response shall be sent when Command message contains invalid parameter. The Parameter Error Location in the Response shall be set to the proper location of the invalid parameter.

References:
NVMe-MI Specification

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Start with Endpoint discovery
2. Send Configuration Get TUSize command to Command Slot 0 with Invalid Config Id = 0x04
3. Wait for response
4. Send NVMe-MI Read VPD with Offset set to more than 256
5. Wait for response
6. Send Read NVMe-MI Data Structure Command with invalid DTYP = 0x05
7. Wait for response

Observable Results:
1. Ensure that Invalid Parameter Status response shall be sent when Command message contains invalid parameter.

Possible Problems: None

Test 4.10 – NVMe-MI Invalid Command Size – (Mandatory)

Purpose: Invalid Command Size Status response should be sent when Command message contains too much/too little data.

References:
NVMe-MI Specification

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform Endpoint Discovery
2. Send Configuration Get TUSize command to Command Slot 0
3. Wait for response
4. Send Configuration Get TUSize command to Command Slot 0 with Length = 12
5. Wait for response

Observable Results:
1. Ensure that Invalid Command size response shall be sent when Command message contains an Invalid Command Size.

Possible Problems: None

Group 5: NVMe Management Interface Tests

Test 5.1 – NVMe-MI Message Type – (Mandatory)

Purpose: The Message Type field specifies the type of payload contained in the message body and is required to be set to 4h in all messages associated with NVMe-MI

References:
NVMe-MI Specification Section 3.2.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Get State Control Primitive Request to the Management Endpoint Command Slot 0
3. Wait for the Response Message

Observable Results:
1. Ensure the Message type in the Response Message is set to 4.
2. If returned correctly, then result is PASS, otherwise, result is FAIL

Possible Problems: None

Test 5.2 – NVMe-MI Message IC – (Mandatory)

Purpose: All NVMe-MI messages are protected by a CRC and thus IC bit shall be set to '1' in all NVMe-MI messages.

References:
NVMe-MI Specification Section 3.2.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization
2. Execute NVMe-MI Configuration Get command
3. Wait for Response
4. Execute NVMe-MI Configuration Get command with IC bit set to zero
5. Wait for Response
6. Execute NVMe-MI Configuration Get command
7. Wait for Response

Observable Results:
1. Ensure that the Configuration Get command returns with successful completion.
2. Ensure that the second Get command with IC bit set to zero is dropped silently.
3. Ensure that the third Get command returns with successful completion.

Possible Problems: None

Test 5.3 – NVMe-MI CRC Check – (Mandatory)

Purpose: The Message Integrity Check field contains a 32-bit CRC computed over the contents of the NVMe-MI message. The 32-bit CRC used by NVMe-MI is CRC-32C (Castagnoli) which uses the generator polynomial 1EDC6F41h

References:
NVMe-MI Specification Section 3.2.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform Endpoint discovery
2. Perform Configuration Get - Transmission Unit Size
3. Wait for response
4. Send Control Primitive with OpCode = Replay
5. Wait for response

Observable Results:
1. Ensure that CRC check passed

Possible Problems: None

Test 5.4 – NVMe-MI Command Slot – (Mandatory)

Purpose: For Response Messages, the CSI field indicates the Command Slot associated with the Request Message with which the Response Message is associated.
(Multi Message checking – use baseline TU Size of 64 and use VPD Read Command – up to 256 bytes)

References:
NVMe-MI Specification Section 3.2.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Get State Control Primitive Request to the Management Endpoint Command Slot 0, wait for response
3. Perform Get State Control Primitive Request to the Management Endpoint Command Slot 1, wait for response
4. Perform Configuration Get MI Command (Cfg ID 0x1) to the Management Endpoint Command Slot 0, wait for response
5. Perform Configuration Get MI Command (Cfg ID 0x1) to the Management Endpoint Command Slot 1, wait for response.

Observable Results:
1. Ensure that for each command the CSI field in the Response Message matches the CSI field in the Request message.
2. If returned correctly, then result is PASS, otherwise, result is FAIL

Possible Problems: None

Test 5.5 –

Test 5.6 – NVMe-MI MCTP packet padding – (Mandatory)

Purpose: The MCTP Transmission Unit size of the last packet in a Request Message or Response Message (i.e., the one with the EOM bit set in the MCTP header) shall be the smallest size needed to transfer the MCTP Packet Payload for that Packet with no additional padding beyond any padding required by the physical medium-specific trailer.

References:
NVMe-MI Specification Section 3.2.2

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get Transmission Unit Size
3. Wait for Response Message;
4. Execute NVMe-MI VPD Read with Data Offset = 0 and Data Length = 256.
5. Wait for Response

Observable Results:
1. Ensure that the Get Endpoint ID command returns with successful completion during initialization.
2. Ensure that the Get Transmission Unit Size returns the default value of 64
3. Ensure that the MCTP packet with EOM = 1 in the VPD Response Message has the smallest size and that the start packet and every other middle packet is bigger than the end packet.

Possible Problems: None

Test 5.7 – NVMe-MI Message Integrity Check – (Mandatory)

Purpose: Once a complete NVMe-MI MCTP message has been assembled, the Message Integrity Check is verified. If the Message Integrity Check passes, then the message is processed. If the Message Integrity Check fails, then the message is discarded

References:
NVMe-MI Specification Section 3.2.2

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform Endpoint Discovery
2. Send Configuration Get TUSize command to Command Slot 0
3. Wait for response
4. Send Configuration Get TUSize command to Command Slot 0
5. Wait for 150 milliseconds to exceed maximum possible latency and ensure that no response was sent
6. Send Configuration Get TUSize command to Command Slot 0
7. Wait for response

Observable Results:
1. Check that message is dropped when MIC fails

Possible Problems: None

Group 6: NVMe-MI Message Processing Tests

Test 6.1 – NVMe-MI Reserved Fields – (Mandatory)

Purpose: The purpose of this test is to check the behavior of the Response Message reserved fields

References:
NVMe-MI Specification Section 6.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Get State Control Primitive Request to the Management Endpoint Command Slot 0, wait for response;
3. Perform Configuration Get MI Command (Cfg ID 0x1) to the Management Endpoint Command Slot 0, wait for response

Observable Results:
1. Ensure that for each command the Reserved fields in the first DWord of the Response Message are set to zero.
2. If returned correctly, then result is PASS, otherwise, result is FAIL

Possible Problems: None

Test 6.2 – NVMe-MI Error Response Code – (Mandatory)

Purpose: The purpose of this test is to check that the Generic Error Response with the proper code is returned for Invalid Command Opcode (0x03), and for Invalid Command Size (0x05) and Invalid Command Input Data Size (0x06)

References:
NVMe-MI Specification Section 6.2

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: July 17, 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform Endpoint Discovery
2. Send normal Configuration Get NVMe-MI command, Cmd slot 0
3. Wait for response
4. Send NVMe-MI command with undefined opcode 0x08
5. Wait for response
6. Send Configuration Get SmBusI2C Freq with Length = 12
7. Wait for response

Observable Results:
1. Check that the Generic Error Response with the proper code is returned for Invalid Command Opcode (0x03), and for Invalid Command Size (0x05) and Invalid Command Input Data Size (0x06)

Possible Problems: None

Group 7: Control Primitives Tests

Test 7.1 – NVMe-MI Response Tag – (Mandatory)

Purpose: Tag in Response matches the Tag in Request

References:
NVMe-MI Specification Section 7.2
MCTP Base Specification Section 8.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Get State Control Primitive Request to the Management Endpoint Command Slot 0, with Tag value 0x01, wait for response
3. Repeat the above for Endpoint Command Slot 1.

Observable Results:
1. Ensure that for each command, proper response is returned, and the Tag value in Response matches the one in Request.
2. If returned correctly, then result is PASS, otherwise, result is FAIL

Possible Problems: None

Test 7.2 – NVMe-MI Response Message – (Mandatory)

Purpose: The intent of this test is to ensure that a DUT returns successful response for the Pause, Resume and Abort Control Primitives.

References:
NVMe-MI Specification Section 7.3

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform Endpoint Discovery
2. Send Control Primitive with OpCode = Pause
3. Wait for response
4. Send Control Primitive with OpCode = Resume
5. Wait for response
6. Send Control Primitive with OpCode = Abort
7. Wait for response

Observable Results:
1. Ensure that a DUT returns successful response for the Pause, Resume and Abort Control Primitives.

Possible Problems: None

Test 7.3 – NVMe-MI Get State Primitive Response – (FYI)

Purpose: Response for the GetState primitive returns the correct state based on the conditions applied. This includes the state error bits and response to Clear Error State Flags

References:
NVMe-MI Specification Section 7.4

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 2, 2019

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get EndPointID command
3. Send Get State Control Primitive Request with CESF = 1
4. Send a second Get State Control Primitive Request with CESF = 1
5. Send Get EndPoint Id with a Bad Header Version
6. Send Get State Control Primitive with CESF = 0
7. Send Get State Control Primitive with CESF = 1
8. Send Get State Control Primitive with CESF = 0
9. Repeat the above with:
10. Send Get EndPoint Id with EOM = 0 and SOM = 0 (Middle Packet)
11. Send Get EndPoint Id with a Destination Endpoint ID = 0x11
12. Send Get EndPoint Id with a Tag Owner = 01
13. Send Configuration Get - MCTP Transmission Unit Size with CRC = 0x123AB
14. Send Configuration Get - MCTP Transmission Unit Size with LCRC = 0xAB (Only for VDM)

Observable Results:
1. Ensure that the Get State Control Primitive Response sent in response to the second Get State Control Primitive Request has bytes 07:08 are all cleared.
2. Ensure that Bit 7 is 1 for Bad Header Version.
3. Ensure that Bit 10 is 1 for the Middle Packet.
4. Ensure that Bit 8 is 1 for Unknown Destination ID.
5. Ensure that Bit 12 is 1 for Bad Tag Owner.
6. Ensure that Bit 4 is 1 for Bad CRC.
7. Ensure that Bit 13 is 1 for Bad LCRC. (Only for VDM)

Possible Problems: None

Test 7.4 – NVMe-MI Response Message Replay – (Mandatory)

Purpose: Verify that DUT replays the response message according to the RRO in request, state and the rules, and that the status and RR bit in the Response control primitive is set according to the state and rules

References:
NVMe-MI Specification Section 7.5

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Execute Get EndPointID command
3. Perform NVMe-MI Configuration Set - Transmission Unit Size
4. Wait for Response
5. Perform NVMe-MI Control Primitive Replay
6. Wait for Response

Observable Results:
1. Ensure that the Configuration Set command returns with success.
2. Ensure that the Replay Control primitive returns with success.
3. -Ensure that bits 01:15 of Byte 07:08 are Reserved
4. Ensure that the Replayed Message was received successfully.

Possible Problems: None

Test 7.5 – NVMe-MI Response Replay Offset (RRO) – (Mandatory)

Purpose: Verify that DUT replays the response message according to the RRO in request

References:
NVMe-MI Specification Section 7.5

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: March 29 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform NVMe-MI VPD Read with Data length set to 256 Bytes.

3. Wait for Response
4. Perform NVMe-MI Control Primitive Replay with Response Replay Offset set to 2.
5. Wait for Response

Observable Results:
1. Ensure that the VPD Read command returns with success.

2. Ensure that the Replay Control primitive returns with success.
3. Ensure that the Replayed Message was received successfully with 3 packets instead of 5 because of RRO.
4. Also ensure that the first packet in the replayed message has SOM set to 1.

Possible Problems: None

Group 8: Management Interface Commands

Test 8.1 – NVMe-MI Response Header – (Mandatory)

Purpose: Reserved field, Integrity Check (verify correct CRC-32)

References:
NVMe-MI Specification Section 4.4

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: July 18, 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization
2. Send an NVMe-MI Configuration Get - Transmission Unit Size

Observable Results:
1. Ensure that a successful Response was returned.
2. Ensure that Byte 3, Byte 2, and bits 1, 2 of Byte 1, are reserved.
3. Ensure that the IC bit is set to 1.

Possible Problems: None

Test 8.2 – NVMe-MI Configuration Set – (Mandatory)

Purpose: Configuration Set.
Specifying a reserved identifier in the Configuration Identifier field causes the command to complete with an Invalid Parameter error status.

References:
NVMe-MI Specification Section 5.2

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization
2. Perform Configuration Set Command with Configuration Identifier 0x03 (MCTP Transmission Unit Size: Size = 64 bytes)
3. Wait for the Response message
4. Perform Configuration Set Command with Configuration Identifier set to 0
5. Wait for the Response message
6. Repeat the above for Configuration Identifier values of 0x04, 0x56, 0xBF

Observable Results:
1. Ensure that the Set Configuration command returns with successful completion.
2. Ensure that for the rest of the commands:
3. In the Parameter Error Location field of the response the byte position is set to 8 and bit position - to 0.

Possible Problems: None

Test 8.3 – NVMe-MI Config Get Response – (Mandatory)

Purpose: Responses to Config Get, Health Status Polls, and VPD Read return data in proper format
Configuration Get:
- SMBus/I2C Frequency. NVMe Management Response Bits 23:4 are reserved, bits 3:0 return the frequency encoding.
- MCTP Transmission Unit Size. NVMe Management Response Bits 23:4 are reserved, bits 3:0 return the unit size.
- Specifying a reserved identifier in the Configuration Identifier field causes the command to complete with an Invalid Parameter error status.

References:
NVMe-MI Specification Section 5.1

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Perform Configuration Get Command requesting MCTP Transmission Unit Size
3. Wait for the Response message
4. Perform Configuration Get Command requesting SMBus/I2C Frequency
5. Wait for the Response message

Observable Results:
1. Ensure that for both commands proper response is returned, and status is set to Success
2. Ensure that for the Command requesting MCTP Transmission Unit Size:
    a. Bits 23:16 of the NVMe Management Response field are reserved (set to zero)
    b. Bits 15:0 return the default unit size of 0x40
3. Ensure that for the Command requesting SMBus/I2C Frequency Unit Size:
    c. Bits 23:4 of the NVMe Management Response field are reserved (set to zero)
    d. Bits 3:0 return the valid frequency encoding (0 to 3)

Possible Problems: None

Test 8.4 – NVMe-MI Health Status Poll – (Mandatory)

Purpose: The intent of this test is to verify the reserved bits of the Controller Health Status Poll Response are set to zero.

References:
NVMe-MI Specification Section 5.4

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. -Perform MCTP Initialization
2. -Send Configuration Get Health Status Poll
3. -Wait for response

Observable Results:
1. Ensure that a proper response is returned with status set to Success
2. Bits 23:16 of dw0 of the NVMe Management response are reserved for all responses
3. Check Bits 15:00 of dw0 for response length for all responses
4. Ensure that for the when DTYP is set to Subsystem Information (0x00)
5. Check that the response length = 12 + 32 = 44 bytes
6. BYTES 31:03 are reserved
7. Ensure that for the when DTYP is set to Port Information (0x01)
8. Check that the response length = 12 + 32 = 44 bytes
9. BYTES 07:04 are reserved
10. BYTE 9 bits 7:3 are reserved
11. BYTES 31-13 are reserved
12. BYTE 01 is reserved
13. Ensure that for the when DTYP is set to Controller List (0x02)
14. Check that the response length = 12 + 8 = 20 bytes (4 bytes per controller)
15. Ensure that for the when DTYP is set to Controller Information (0x03)
16. Check that the response length = 12 + 32 = 44 bytes
17. BYTES 7:1 are reserved
18. BYTES 4-1 are reserved
19. BYTE 5 bits 7:1 are reserved
20. BYTES 31:16 are reserved
21. Ensure that for the when DTYP is set to optional Commands supported (0x04)
22. Check that the response length = 12 + 6 = 18 bytes (CMD0 and CMD1)
23. bits 2:0 and bit 7 of BYTE 00 are reserved for each entry in the list

Possible Problems: None

Test 8.5 – NVMe-MI Controller Health Status Poll – (FYI)

Purpose: Controller Health Status Poll:
- The length should correspond to RENT in Management response
- For each Controller Health data structure bytes 15:9 and bits 15:8 in bytes 322 are reserved

References:
NVMe-MI Specification Section 5.3

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: November 26, 2018

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A. Case 1: NVMe-MI 1.0 ECN 003 Not Implemented (M)

Case 1: NVMe-MI 1.0 ECN 003 or NVMe-MI 1.0a or higher Not Implemented (M)

Test Procedure:
1. The following procedure is only applicable to a device implementing NVMe-MI 1.0 which has not implemented NVMe.MI 1.0 ECN003.
2. Perform MCTP initialization.
3. Perform Controller Health Status Poll Command with the following parameters:Report all set, Include PCI Functions set, Controller Status Changes set,MAXRENT set to 0x01, all other parameters set to zero.Wait for the Response message

Observable Results:
1. Ensure that the proper response is returned, and status is set to Success
2. Ensure that for the following bits are reserved in the response, and cleared to 0.
    a. Bits 15:8 in DWORD 1
3. Bits 31:24 in DWORD 2Bits 31:5 in DWORD 4Bits 31:0 in DWORD 5Ensure that if a Controller Health Data Structure is included in the response, that RENT (bits 23:16 in DWORD 1) use a 0's based value (i.e a value of 0 indicated 1 Data Structure included).

Case 2: NVMe-MI 1.0 ECN 003 or NVMe-MI 1.0a or higher Implemented (M)

Test Procedure:
1. The following procedure is only applicable to a device implementing NVMe.MI 1.0 which has implemented ECN003 and to devices will implement NVMe-MI 1.0a or higher.
2. Perform MCTP initialization
3. Perform Controller Health Status Poll Command with the following parameters: Report all set, Include PCI Functions set, Controller Status Changes set, MAXRENT set to 0x01, all other parameters set to zero.
4. Wait for the Response message

Observable Results:
1. Ensure that the proper response is returned, and status is set to Success
2. Ensure that for the following bits are reserved in the response, and cleared to 0.
    a. Bits 15:8 in DWORD 1
    b. Bits 31:24 in DWORD 2
    c. Bits 31:5 in DWORD 4
    d. Bits 31:0 in DWORD 5

3. Ensure that if a Controller Health Data Structure is included in the response, that RENT (bits 23:16 in DWORD 1) is set to a non-zero value.

Possible Problems: NVMe-MI 1.0 ECN 003 changed the RENT field from being a zeroes based value to not being a zeroes based value. Products implementing NVMe-MI 1.0 may or may not implement NVMe-MI 1.0 ECN 003. Products implementing NVMe-MI 1.0a or higher must implement NVME-MI 1.0 ECN 003.

Test 8.6 – NVMe-MI Read Data Structure – (Mandatory)

Purpose:
The intent of this test is to verify the response length and reserved bits of the Get Read Subsystem information response for different Data Structure Types.
References:
NVMe-MI Specification Section 5.5

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP initialization
2. Send Configuration Read NVMe-MI Data Structure : with DTYP = 0x00
3. Send Configuration Read NVMe-MI Data Structure : with DTYP = 0x01
4. Send Configuration Read NVMe-MI Data Structure : with DTYP = 0x02
5. Send Configuration Read NVMe-MI Data Structure : with DTYP = 0x03
6. Send Configuration Read NVMe-MI Data Structure : with DTYP = 0x04

Observable Results:
1. Ensure that the proper response is returned, and status is set to Success
2. Ensure that for the following bits are reserved in the response.
    a. Bits 23:0 in DWORD 1
    b. Bits 1:0 and 7:6 in DWORD 2
    c. Bits 31:13 in DWORD 3

Possible Problems: None

Test 8.7 – NVMe-MI Data Length – (Mandatory)

Purpose: The intent of this test is to verify the response length of specific requested parameters.

References:
NVMe-MI Specification Section 5.5

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: November 26, 2018

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Case 1: Verify NVMSSI Data Length (M)

Test Procedure:
1. Perform Endpoint Discovery
2. Send Configuration Read NVMe-MI Data Structure: with DTYP = 0x00

Observable Results:
1. Verify the response length is 32

Case 2: Verify PortInfo Data Length (M)

Test Procedure:
1. Perform Endpoint Discovery
2. Send Configuration Read NVMe-MI Data Structure: with DTYP = 0x01

Observable Results:
1. Ensure Response Data Length is 32

Case 3: Verify CtrlrList Data Length (M)

Test Procedure:
1. Perform MCTP Initialization
2. -Perform Read NVMe-MI Data Structure with DTYP set to 0x02 (Controller List)
3. -Wait for Response Message

Observable Results:
1. Ensure that a proper response is returned with status set to Success
    a. Bits 23:16 of dw0 of the response are reserved for all responses
    b. Check Bits 15:00 of dw0 for response length for all responses
    c. Ensure that for the when DTYP is set to Controller List (0x02)
2. Check that the response length = 12 + 8 = 20 bytes (4 bytes per controller)

Case 4: Verify CtrlrInfo Data Length (M)

Test Procedure:
1. -Perform MCTP Initialization
2. -Perform Read NVMe-MI Data Structure with DTYP set to 0x03 (Controller Information)
3. -Wait for Response Message

Observable Results:

1. Ensure that a proper response is returned with status set to Success
2. Bits 23:16 of dw0 of the response are reserved for all responses
3. Check Bits 15:00 of dw0 for response length for all responses
4. Ensure that for the when DTYP is set to Controller Information (0x03)
5. Check that the response length = 12 + 32 = 44 bytes
6. BYTES 4:1 are reserved
7. Bits 7:1 of byte 5
8. BYTES 31:16 are reserved

Case 5: Verify OptCmds Data Length (M)

Test Procedure:

1. -Perform MCTP Initialization
2. -Perform Read NVMe-MI Data Structure with DTYP set to 0x04 (Optional Commands)
3. -Wait for Response Message

Observable Results:

1. Ensure that a proper response is returned with status set to Success
2. Bits 23:16 of dw0 of the response are reserved for all responses
3. Check Bits 15:00 of dw0 for response length for all responses
4. Ensure that for the when DTYP is set to optional Commands supported (0x04)
5. Check that the response length = 12 + 6 = 18 bytes (CMD0 and CMD1)
6. Bits 2:0 and bit 7 of BYTE 00 are reserved

Possible Problems: None

Test 8.7.2

Group 9: NVMe Admin Command Set Tests

Test 9.1 – NVMe-MI Mandatory Admin Commands – (In Progress)

Purpose: Check operation of the 3 Mandatory Admin Commands

References:
NVMe-MI Specification

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. TBD

Observable Results:
1. TBD

Possible Problems: None

Group 10: Management Enhancement Tests

Test 10.1 – NVMe-MI Identify Structure ME Capabilities – (In Progress)

Purpose: New Identify Controller structure fields (ME Capabilities)

References:
TBD

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
TBD

Observable Results:
TBD

Possible Problems: None

Test 10.2 – NVMe-MI Identify Capabilities – (In Progress)

Purpose: In NVMe Identify Controller data structure bytes 254:240 and byte 255 bits 7:2 are reserved. Bits 0 and 1 in byte 255 identify presence of the Management Endpoint on the port (SMBus/I2c and PCIe) – should probably use this information from the start to decide what can be tested for the port.

References:
TBD

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
TBD

Observable Results:
TBD

Possible Problems: None

Test 10.3 – NVMe-MI Namespace Metadata – (In Progress)

Purpose: Test Adding, Updating and Deleting Controller and Namespace Metadata Elements.

References:
TBD

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: April 20 2016

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
TBD

Observable Results:
TBD

Possible Problems: None

Group 11: Vital Product Data Tests

Test 11.1 – NVMe-MI VPD Default Values – (FYI)

Purpose: Factory default values / reserved fields set properly

References:
TBD

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

Test Procedure:
1. Perform MCTP Initialization
2. Perform VPD Read
3. Wait for Response

Observable Results:
1. Ensure that VPD Read returned with Success.
4. Ensure that the Following Bits were reserved in the Response.
5. Common Header:
6. Byte 01 = 0x01 (IPMIVER)
7. Byte 04 = 0x01 (PIAOFF)
8. Byte 05 = 0x0F (MRIOFF)
9. Byte 06 = 0x00 (Reserved)
10. Product Info Area:
11. Byte 00 = 0x01 (IPMIVER)
12. Byte 01 = 0x0E (PALEN)
13. Byte 03 = 0xC8 (MNTL)
14. Byte 12 = 0xD8 (PNTL)
15. Byte 37 = 0xE8 (PPMNTL)
16. Byte 78 = 0xC2 (PVTL)
17. Byte 81 = 0xD4 (PSNTL)
18. Byte 104 = 0xC1 (EOR)
19. Byte 110:105 = 0x00 (Reserved)

Possible Problems: None

Group 12: Management Endpoint Reset Tests

Test 12.1 – NVMe-MI PCIe Reset – (FYI)

Purpose: PCIe resets leading to ME reset

References:

Resource Requirements:
Tools capable of stimulus and decoding of traffic on the MCTP and NVMe-MI interface.

Last Modification: February 15 2017

Discussion: No further discussion available at time of writing

Test Setup: See Appendix A.

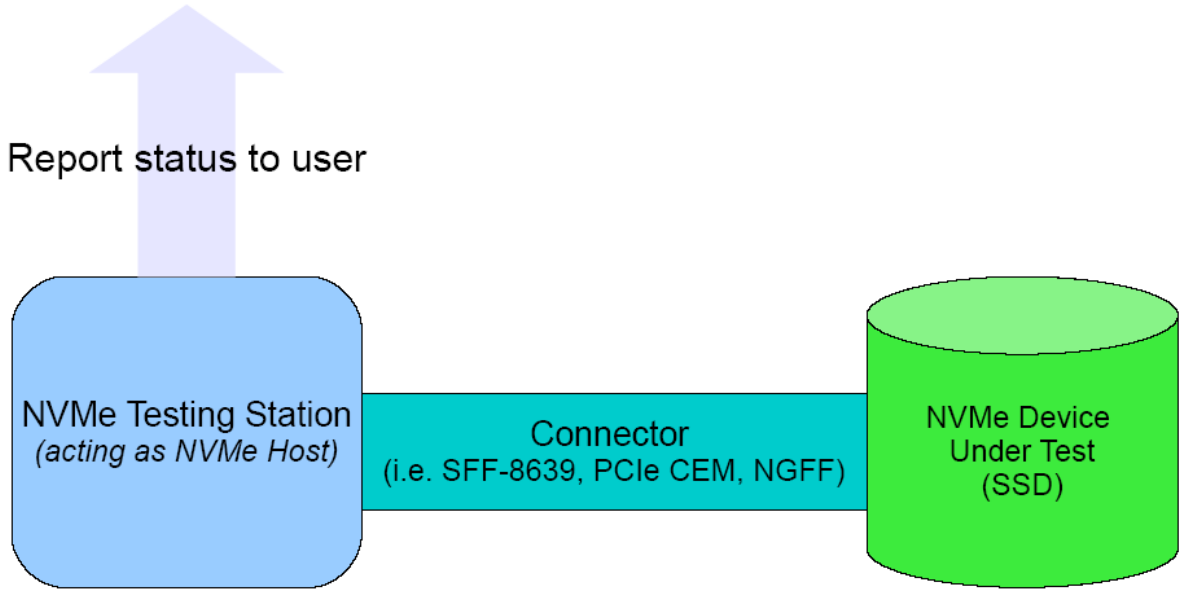
Test Procedure:
TBD

Observable Results:
TBD

Possible Problems: None

Appendix A: Default Test Setup

Except where otherwise specified, all tests will require the DUT to have one of the following default physical configuration at the beginning of each test case:



Appendix B: Notes on Test Procedures

There are scenarios where in test procedures it is desirable to leave certain aspects of the testing procedure as general as possible. In these cases, the steps in the described test procedure may use placeholder values, or may intentionally use non–specific terminology, and the final determination of interpretation or choice of values is left to the discretion of the test technician. The following is an attempt to capture and describe all such instances used throughout the procedures.

Ports on Testing Station and Device Under Test

In general, any PCIe Port on the Testing Station or Device Under Test may be used as an interface with a test station or interoperability partner. There is *assumed* to be no difference in behavior, with respect to the protocols involved in this test suite, between any two PCIe ports on the Testing Station or Device Under Test. Hence, actual ports used may be chosen for convenience. However, it is recommended that the PCIe port used in the test configuration is recorded by the test technician.

Use of "various"

To maintain generality, some steps will specify that "various other values" (or the like) should be used in place of a given parameter. Ideally, all possible values would be tested in this case. However, limits on available time may constrain the ability of the test technician to attempt this. Given this, a subset of the set of applicable values must generally be used.
When deciding how many values should be used, it should be noted that the more values that are tested, the greater the confidence of the results obtained (although there is a diminishing return on this).
When deciding which specific values to use, it is generally recommended to choose them at pseudo–randomly yet deterministically. However, if there exist subsets of the applicable values with special significance, values from each subset should be attempted.

Appendix C: NVMe Integrators List Requirements

Purpose: To provide guidance on what tests are required for NVMe Integrators List Qualification for products that advertise support for NVMe-MI on the NVMe Integrators List.

References:
UNH-IOL NVMe Integrators List Policy Document
NVMe-MI Conformance Test Specification

Resource Requirements:
NVMe-MI Device.

Last Modification: April 29, 2019

Discussion: The table below outlines what tests are to be considered Mandatory, FYI, or In Progress for devices being qualified for the NVMe Integrators List that advertise support for NVMe-MI on the NVMe Integrators List. Further descriptions of each test can be found in the NVMe-MI Conformance Test Specification.

MCTP Base Tests

Test Name and Number	Mandatory, FYI, In Progress?
Test 1.1 – MCTP Endpoint ID	FYI
Test 1.1.1 – MCTP Endpoint ID – Reserved Bits	FYI
Test 1.2 – MCTP Packet Sequence Number	FYI
Test 1.3 – MCTP Multiple Packets	FYI
Test 1.4.1 – MCTP Bad Packet 1	FYI
Test 1.4.1 – MCTP Bad Packet 2	FYI
Test 1.4.1 – MCTP Bad Packet 3	FYI
Test 1.4.1 – MCTP Bad Packet 4	FYI
Test 1.4.1 – MCTP Bad Packet 5	FYI

MCTP Control Message Tests

Test Name and Number	Mandatory, FYI, In Progress?
Test 2.1 – MCTP Control Instance ID	FYI

MCTP Commands

Test Name and Number	Mandatory, FYI, In Progress?
Test 3.1 – MCTP Command Set Endpoint ID	FYI
Test 3.2 – MCTP Command Get MCTP Version	FYI
Test 3.3 – MCTP Command Get Message Type	FYI
Test 3.4 – MCTP Command Prepare for Endpoint Discovery	FYI
Test 3.5 – MCTP Command Endpoint Discovery	FYI
Test 3.6 – MCTP Command Get Endpoint ID	FYI

NVMe Error Handling

Test Name and Number	Mandatory, FYI, In Progress?
Test 4.1 - NVMe-MI Invalid Opcode	Mandatory
Test 4.2 - NVMe-MI Reserved Identifier	Mandatory
Test 4.3 - NVMe-MI Health Status Change	Mandatory
Test 4.4 - NVMe-MI Reserved Identifier 2	Mandatory
Test 4.5 - NVMe-MI MAXRENT Error	Mandatory
Test 4.6 - NVMe-MI Reserved Data Structure Type	Mandatory

Test 4.7 - NVMe-MI Invalid VPD Read Size	FYI
Test 4.8 - NVMe-MI Invalid VPD Write Status	Mandatory
Test 4.9 - NVMe-MI Invalid Parameter Status	Mandatory
Test 4.10 - NVMe-MI Invalid Command Size	FYI

NVMe Management Interface Tests

Test Name and Number	Mandatory, FYI, In Progress?
Test 5.1 – NVMe-MI Message Type	Mandatory
Test 5.2 – NVMe-MI Message IC	Mandatory
Test 5.3 – NVMe-MI CRC Check	FYI
Test 5.4 – NVMe-MI Command Slot	Mandatory
Test 5.5 – NVME-MI Request Data size mismatch	Mandatory
Test 5.6 – NVMe-MI MCTP packet padding	Mandatory
Test 5.7 – NVMe-MI Message Integrity Check	Mandatory

NVMe-MI Message Processing Tests

Test Name and Number	Mandatory, FYI, In Progress?
Test 6.1 – NVMe-MI Reserved Fields	Mandatory
Test 6.2 – NVMe-MI Error Response Code	Mandatory

Control Primitives Tests

Test Name and Number	Mandatory, FYI, In Progress?
Test 7.1 – NVMe-MI Response Tag	Mandatory
Test 7.2 – NVMe-MI Response Message	Mandatory
Test 7.3 – NVMe-MI GetState Primitive Response	FYI
Test 7.4 – NVMe-MI Response Message Replay	Mandatory

Management Interface Commands

Test Name and Number	Mandatory, FYI, In Progress?
Test 8.1 – NVMe-MI Response Header	Mandatory
Test 8.2 – NVMe-MI Configuration Set	Mandatory
Test 8.3 – NVMe-MI Config Get Response	Mandatory
Test 8.4 – NVMe-MI Health Status Poll	Mandatory
Test 8.5 – NVMe-MI Controller Health Status Poll	Mandatory
Test 8.6 – NVMe-MI Read Data Structure	Mandatory
Test 8.7.1 – NVMe-MI NVMSSI Data Length	Mandatory
Test 8.7.2 – NVMe-MI PortInfo Data Length	Mandatory
Test 8.7.3 – NVMe-MI CtrlrList Data Length	Mandatory
Test 8.7.4 – NVMe-MI CtrlrInfo Data Length	Mandatory
Test 8.7.5 – NVMe-MI OptCmds Data Length	Mandatory

NVMe Admin Commands Set Tests

Test Name and Number	Mandatory, FYI, In Progress?
Test 9.1 – NVMe-MI Mandatory Admin Commands	In Progress

Management Enhancement Tests

Test Name and Number	Mandatory, FYI, In Progress?
Test 10.1 – NVMe-MI Identify Structure ME Capabilities	In Progress
Test 10.2 – NVMe-MI Identify Capabilities	In Progress
Test 10.3 – NVMe-MI Namespace Metadata	In Progress

Vital Product Data Tests

Test Name and Number	Mandatory, FYI, In Progress?
Test 11.1 – NVMe-MI VPD Default Values	In Progress

Management Endpoint Reset Tests

Test Name and Number	Mandatory, FYI, In Progress?
Test 12.1 – NVMe-MI PCIe Reset	In Progress

Appendix D: TEST TOOLS

The Tests described in this document can be performed using available Teledyne-LeCroy Test Tools. These tests are supported in v8.77 or higher of the Teledyne-LeCroy PC Edition software available at:
http://teledynelecroy.com/support/softwaredownload/documents.aspx?standardid=18