UNH–IOL
NVMe Testing Service

Test Plan for NVMe PCIe Transport Conformance
Version 18.0
Target Specification: NVMe over PCIe Transport Specification 1.0a
Technical Document

Last Updated: July 14, 2022

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

2022 January 21 (Version 1.0) Initial Release

Tim Sheehan:

1. Initial release of the NVMe PCIe Transport Conformance Test Suite v17, inclusive of PCIe tests from the NVMe Conformance Test Suite v16.0

2022 July 14 (Version 18.0) Final Release

Tim Sheehan:

1. Program Revision Update

ACKNOWLEDGMENTS

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

Tim Sheehan UNH InterOperability Laboratory

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 functionality of their products. This test suite is aimed at validating products in support of the work being directed by the NVMe Promoters Group.

These tests are designed to determine if a product conforms to specifications defined in the NVM Express PCIe Transport Specification Revision 1.0 specification and NVM Express Base Specification 2.0a. Successful completion of these tests provide a reasonable level of confidence that the Device Under Test (DUT) will function properly in many NVMe environments. Transport testing should also include those tests for tests in respective I/O command set test plans

The tests contained in this document are organized in order to simplify the identification of information related to a test, and to facilitate in the actual testing process. Tests are separated into groups, primarily in order to reduce setup time in the lab environment, however the different groups typically also tend to focus on specific aspects of device functionality. A two–number, dot–notated naming system is used to catalog the tests. This format allows for the addition of future tests in the appropriate groups without requiring the renumbering of the subsequent tests. The NVMe 2.0 refactoring effort has made it important to reference pre-2.0 test cases that have moved to a 2.0 refactored test plan. This new test case numbers will be realized at the end of each test case name with the legacy test case number noted.

The test definitions themselves are intended to provide a high–level description of the motivation, resources, procedures, and methodologies specific to each test. Formally, each test description contains the following sections:

Purpose
The purpose is a brief statement outlining what the test attempts to achieve. The test is written at the functional level.

References
This section specifies all reference material *external* to the test suite, including the specific references for the test in question, and any other references that might be helpful in understanding the test methodology and/or test results. External sources are always referenced by a bracketed number (e.g., [1]) when mentioned in the test description. Any other references in the test description that are not indicated in this manner refer to elements within the test suite document itself (e.g., "Appendix 5.A", or "Table 5.1.1–1").

Resource Requirements
The requirements section specifies the test hardware and/or software needed to perform the test. This is generally expressed in terms of minimum requirements, however in some cases specific equipment manufacturer/model information may be provided.

Last Modification
This specifies the date of the last modification to this test.

Discussion
The discussion covers the assumptions made in the design or implementation of the test, as well as known limitations. Other items specific to the test are covered here as well.

Test Setup
The setup section describes the initial configuration of the test environment. Small changes in the configuration should not be included here, and are generally covered in the test procedure section (next).

Procedure

The procedure section of the test description contains the systematic instructions for carrying out the test. It provides a cookbook approach to testing, and may be interspersed with observable results. These procedures should be the ideal test methodology, independent of specific tool limitations or restrictions.

Observable Results
This section lists the specific observable items that can be examined by the tester in order to verify that the DUT is operating properly. When multiple values for an observable are possible, this section provides a short discussion on how to interpret them. The determination of a pass or fail outcome for a particular test is generally based on the successful (or unsuccessful) detection of a specific observable.

Possible Problems
This section contains a description of known issues with the test procedure, which may affect test results in certain situations. It may also refer the reader to test suite appendices and/or other external sources that may provide more detail regarding these issues.

REFERENCES

The following documents are referenced in this text:

Old Ref : NVM Express Revision 1.4 Specification (June 10, 2019)

1. NVM Express Base Specification Revision 2.0a (July 23, 2021)
2. NVMe over PCIe Transport Specification version 1.0a (July 23, 2021)

ABBREVIATIONS

The following abbreviations are applied to the test titles of each of the tests described in this document for indicating the status of test requirements.

M - Mandatory
FYI - FYI
IP - In Progress

The following abbreviations applied to the test titles of each of the tests described in this document for indicating what product types a test may apply to. It is assumed that all tests apply to base NVMe products using PCIe.

OF – Test applies to NVMe-oF products

The following legacy numbering is applied to the test titles of each of the test cases to map the origin of the test case movement due to 2.0 refactoring efforts. If there is no change to the test case number then this field will be left off of the 2.0 refactored test case title.

NC – UNH-IOL NVMe Conformance Test Plan version 16.0, September 23, 2021
Legacy numbering : (PCIeC:x.x)

Group 1: Controller Architecture (PCIeC Group 6)

Overview:

This section describes a method for performing conformance verification for NVMe products implementing the NVMe Controller Architecture.

Notes:

The preliminary draft descriptions for the tests defined in this group are considered complete, and the tests are pending implementation (during which time additional revisions/modifications are likely to occur).

Test 1.1 – Controller Level Reset – Conventional Reset (M) (PCIeC 6.1)

Purpose: To verify that an NVMe Controller performs the proper actions when a Conventional Reset occurs.

References:

Old Ref : NVMe Specification 7.3.2
NVM Express Base specification 2.0a : 3.7.2
NVMe over PCIe Transport Specification 1.0a : 3.3.1

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: November 26, 2018

Discussion: When a Controller Level Reset occurs, the Host and Controller are required to take specific action.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Device to perform a Conventional Controller Level Reset. Repeat this for each of the following cases:
    a. PCI Express Hot reset
2. When the reset is complete, configure the NVMe Host to issue a Write and then a Read command to the NVMe Controller.

Observable Results:

1. Verify that the NVMe Controller is able to properly execute NVMe Write and Read commands after the reset is complete.
2. Verify that the NVMe Controller performs the following actions when each reset case defined above is initiated:
    a. The controller stops processing any outstanding Admin or I/O commands.
    b. All I/O Submission Queues are deleted.
    c. All I/O Completion Queues are deleted.
    d. The controller is brought to an Idle state. When this is complete, CSTS.RDY is cleared to '0'.
    e. All controller registers defined in section 3 of the NVMe Specification and internal controller state are reset.

Possible Problems: None.

Test 1.2 – Controller Level Reset – Function Level Reset (M) (PCIeC 6.2)

Purpose: To verify that an NVMe Controller performs the proper actions when a Function Level Reset occurs.

References:

Old Ref : NVMe Specification 7.3.2
NVMe over PCIe Transport Specification 1.0a : 3.3.1

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: March 16, 2016

Discussion: When a Controller Level Reset occurs, the Host and Controller are required to take specific action.

Support for the Function Level Reset mechanism is indicated in the Function Level Reset Capability (FLRC) field (bit 28) of the PCI Express Device Capabilities (PXDCAP) (offset PXCAP + 4h) PCI Express Register. All NVMe Controllers must support Function Level Reset and so this register value should be set to '1' for all NVMe Controllers.

A Function Level Reset is initiated by the NVMe Host by writing a value of '1' to the Initiate Function Level Reset field (bit 15) of the PCI Express Device Control (PXDC) (offset PXCAP + 8h) PCI Express Register. The value read by software from this field shall always be '0'.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the PXDCAP.FLRC PCI Express register field.
2. Configure the NVMe Host to read the Initiate Function Level Reset field of the PXDC PCI Express register.
3. Configure the NVMe Host to write a value of '1' to the Initiate Function Level Reset field of the PXDC PCI Express register in order to initiate a Function Level Reset.
4. When the reset is complete, configure the NVMe Host to issue a Write command to the NVMe Controller.

Observable Results:

1. Verify that the value read from the PXDCAP.FLRC PCI Express register field is '1' to indicate support for the Function Level Reset mechanism.
2. Verify that the value read from the Initiate Function Level Reset field of the PXDC PCI Express register is '0'.
3. Verify that the NVMe Controller is able to properly execute the Write command after the reset is complete.
4. Verify that the NVMe Controller performs the following actions when the Function Level Reset is initiated:
    a. The controller stops processing any outstanding Admin or I/O commands.
    b. All I/O Submission Queues are deleted.
    c. All I/O Completion Queues are deleted.
    d. The controller is brought to an Idle state. When this is complete, CSTS.RDY is cleared to '0'.
    e. All controller registers defined in section 3 of the NVMe Specification and internal controller state are reset.

Possible Problems: A reliable means of performing this test has not been determined. Therefore, this test should not be included in any industry approved determination of conformance.

Test 1.3 – Controller Level Reset – NVM Subsystem Reset (M) (PCIeC 6.4)

Purpose: To verify that an NVMe Controller performs the proper actions when an NVM Subsystem Reset occurs.

References:

Old Ref : NVMe Specification 7.3.1 and 7.3.2
NVMe over PCIe Transport Specification 1.0a : 3.3.1

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: April 2, 2019

Discussion: When a Controller Level Reset occurs, the Host and Controller are required to take specific action.

An NVM Subsystem Reset is initiated when:

1. Power is applied to the NVM System,
2. A value of 4E564D65h ("NVMe") is written to the NSSR.NSSRC controller register field, or
3. A vendor specific event occurs.

When an NVM Subsystem Reset occurs, the entire NVM subsystem is reset. This includes the initiation of a Controller Level Reset on all controllers that make up the NVM subsystem and a transition to the Detect LTSSM state by all PCI Express ports of the NVM subsystem.

The occurrence of an NVM Subsystem Reset while power is applied to the NVM subsystem is reported by the initial value of the CSTS.NSSRO field following the NVM Subsystem Reset. This field may be used by host software to determine if the sudden loss of communication with a controller was due to an NVM Subsystem Reset or some other condition.

This test is only applicable if the CAP.NSSRS field is set '1' to indicate support for writing to the NSSR.NSSRC field.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CAP.NSSRS field to determine if the DUT supports writing to the NSSR.NSSRC field. If the NSSRS field is cleared to '0', the test is not applicable. If the CAP.NSSRS field is set to '1', continue to step 2.
2. Configure the NVMe host to read the CSTS.NSSRO register value, and record the value. If the value is not zero, perform a power cycle of the DUT and restart the test.
3. Configure the NVMe host to write a value of 4E564D65h ("NVMe") to the NSSR.NSSRC field.
4. When the reset is complete, the PCIe link is reestablished, the NVMe controller is enabled and an Identify is performed.

Observable Results:

1. Verify that when the reset is performed, the drive is temporarily no longer visible from the host system. Depending on the reset and bring up speed of the device, this may not be observable.
2. Verify that the NVMe Controller is able to properly execute NVMe Identify command after the reset is complete.
3. Verify that the NVMe Controller performs the following actions when the NVM Subsystem Reset is initiated:
    a. The controller stops processing any outstanding Admin or I/O commands.
    b. All I/O Submission Queues are deleted.
    c. All I/O Completion Queues are deleted.
    d. The controller is brought to an Idle state. When this is complete, CSTS.RDY is cleared to '0'.

e. All controller registers defined in section 3 of the NVMe Specification and internal controller state are reset.

Possible Problems: The DUT may or may not support NVM Subsystem Reset as it is an optional NVMe feature. When the reset is performed, the drive is temporarily no longer visible from the host system. Depending on the reset and bring up speed of the device, this may not be observable.

Group 2: Power State Transitions

Overview:

This section describes a method for performing conformance verification for NVMe products implementing NVMe Power State Transitions.

Notes:

The preliminary draft descriptions for the tests defined in this group are considered complete, and the tests are pending implementation (during which time additional revisions/modifications are likely to occur).

Test 2.1 – Autonomous Power State Transitions Enabled (M) (PCIeC 8.1)

Purpose: To determine if an NVMe Controller properly supports Autonomous Power State Transitions.

References:

Old Ref : NVMe Specification 5.14.1.12, Fig 90, Fig 108, Fig 124
NVM Express Base specification 2.0a : 5.27.1.19
NVMe over PCIe Transport Specification 1.0a : 3.6 & 3.8.2.3

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: April 13, 2015

Discussion: An NVMe device may change power states autonomously without host intervention if Autonomous Power State Transitions are supported by the device and enabled by the Host. There are 32 allowable power states defined sequentially in the 256 byte data structure entry.

Each entry is 64 bits long and describes the Idle Time Prior to Transition (ITPTT) and Idle Transition Power State (ITPS). The Idle Transition Power State specifies the next power state the device will transition to after there is a continuous period of idle time in the current power state that exceeds the time specified in the Idle Time Prior to Transition field.

This test is not applicable to devices that do not claim to support Autonomous Power State Transitions.

Test Setup: See Appendix A.

Test Procedure:

1. Check that the DUT supports Autonomous Power State Transitions by setting Bit 0 of Byte 265 of the Identify Controller Data Structure, to 1. If this bit is set to 0, the test is not performed.
2. Using the Set Features Command, enable Feature Identifier 0Ch for Autonomous Power State Transition.
3. Perform the Get Feature Command to see that the Autonomous Power State Transitions feature was enabled (APSTE), and receive the Autonomous Power State Transition data structure.

Observable Results:

1. Verify that the controller returns a properly formatted Autonomous Power State Transition data structure.

Possible Problems: A reliable means of performing this test has not been determined. Therefore, this test should not be included in any industry approved determination of conformance.

Test 2.2 – Return from Non–Operational State (FYI) (PCIeC 8.2)

Purpose: To determine if an NVMe Controller properly supports Autonomous Power State Transitions.

References:

Old Ref : NVMe Specification 8.4.1
NVM Express Base specification 2.0a : 5.27.1.19
NVMe over PCIe Transport Specification 1.0a : 3.6 & 3.8.2.3

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: January 23, 2018

Discussion: When in a non–operational power state, regardless of whether autonomous power state transitions are enabled, the controller shall autonomously transition back to the last operational power state when an I/O Submission Queue Tail Doorbell is written.

When in a non-operational power state, regardless of whether autonomous power state transitions are enabled, the controller shall not transition back if an Admin Command has been sent.
Test Setup: See Appendix A.

Test Setup: See Appendix A.

Case 1: Basic Operation (FYI) (PCIeC 8.2.1)

Test Procedure:

1. Check that the DUT supports Autonomous Power State Transitions by setting Bit 0 of Byte 265 of the Identify Controller Data Structure, to 1. If this bit is set to 0, the test is not performed.
2. Using the Set Features Command, enable Feature Identifier 0Ch for Autonomous Power State Transition.
3. Perform the Get Feature Command to see that the Autonomous Power State Transitions feature was enabled (APSTE), and receive the Autonomous Power State Transition data structure.
4. If the Autonomous Power State Transition Data Structure indicates that the device supports entering a non–operational state via APST, allow the DUT to remain idle for ITPT for each power state successively until the DUT enters a non–operational state.
5. Perform Identify Power State Descriptor Data Structure, check the NOPS field.
6. Perform an NVMe I/O Command, such as NVMe Write to the DUT.
7. Perform Identify Power State Descriptor Data Structure, check the NOPS field.

Observable Results:

1. Verify that the controller returns a properly formatted Autonomous Power State Transition data structure.

2. Verify that after the writing of the I/O Submission Queue Tail Doorbell, through the write command, the DUT returns to the last operational power state.

Case 2: Non-Operation State Admin Commands (FYI) (PCIeC 8.2.2)

Test Procedure:

1. Check that the DUT supports Autonomous Power State Transitions by setting Bit 0 of Byte 265 of the Identify Controller Data Structure, to 1. If this bit is set to 0, the test is not performed.
2. Using the Set Features Command, enable Feature Identifier 0Ch for Autonomous Power State Transition.
3. Perform the Get Feature Command to see that the Autonomous Power State Transitions feature was enabled (APSTE), and receive the Autonomous Power State Transition data structure.

4. If the Autonomous Power State Transition Data Structure indicates that the device supports entering a non-operational state via APST, allow the DUT to remain idle for ITPT for each power state successively until the DUT enters a non-operational state.
5. For each supported Admin Command opcode, preform an Admin Command of that opcode, then perform Identify Power State Descriptor Data Structure, check the NOPS field.

Observable Results:
1. Verify that the controller returns a properly formatted Autonomous Power State Transition data structure.
2. Verify that after each Admin Command, the DUT stays in the Non-Operational Power State.

Possible Problems: None.

Test 2.3 – Autonomous Power State Transition (M) (PCIeC 8.3)

Purpose: To verify that an NVMe system can properly handle autonomous power state transitions.

References:

Old Ref : NVMe Specification 8.4.2, 5.14.1.12
NVM Express Base specification 2.0a : 5.27.1.19
NVMe over PCIe Transport Specification 1.0a : 3.6 & 3.8.2.3

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: April 1, 2016

Discussion: The controller may support autonomous power state transitions, as indicated in the Identify Controller data structure at byte 265. Autonomous power state transitions provide a mechanism for the host to configure the controller to automatically transition between power states on certain conditions without software intervention.

The entry condition to transition to the Idle Transition Power State is that the controller has been in idle for a continuous period of time exceeding the Idle Time Prior to Transition time specified. The controller is idle when there are no commands outstanding to any I/O Submission Queue. The power state to transition to shall be a non-operational power state (a non-operational power state may autonomously transition to another non-operational power state). If an operational power state is specified then the controller should abort the command with a status of Invalid Field in Command.

Test Setup: See Appendix A

Case 1: Proper Structure

Discussion: Each entry in the Autonomous Power State Transition data structure is defined in the NVMe Specification. Each entry is 64 bits in size. There is an entry for each of the allowable 32 power states. For power states that are not supported, the unused Autonomous Power State Transition data structure entries shall be cleared to all zeroes. The entries begin with power state 0 and then increase sequentially (i.e., power state 0 is described in bytes 7:0, power state 1 is described in bytes 15:8, etc.). The data structure is 256 bytes in size and shall be physically contiguous.

Test Procedure:

1. Configure the NVMe Host to issue a Set Features command with an unsupported power state.
2. Configure the NVMe Host to issue a Get Features command with the feature ID set to Autonomous Power State Transition (0Ch) to the NVMe Controller.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify that the Autonomous Power State Transition data structure and each of its entries are of proper size.
3. Verify that Autonomous Power State Transition data structure entries for power states not supported by the controller are cleared to all zeroes.

Test 2.4 – Power State Entrance Latency (M) (PCIeC 8.4)

Purpose: To verify that an NVMe system properly documents its entrance latency for each Power State.

Reference:

Old Ref : NVMe Specification 8.4
NVM Express Base specification 2.0a : 8.15
NVMe over PCIe Transport Specification 1.0a : 3.6

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: January 23, 2018

Discussion

The controller may support a Power States Entrance Latency field, as indicated in the Power State Descriptor table described in the NVMe specification. This field indicates how long it should take, in microseconds, for a controller's power state to transition to another.

Test Setup: See Appendix A

Test Procedure:

1. For each Power State in the Power State Descriptor table with an ENLAT field that is not 0h:
2. Record the ENLAT field of the selected Power State and the EXLAT field of the current Power State
3. Have the NVMe host send a Set Feature Command with Feature Identifier 02h, Power Management, indicating the new Power State
4. Wait the EXLAT of the previous Power State and the ENLAT of the power state being selected.
5. Send a Get Features command with Feature Identifier 02, Power Management.

Observable Results:

1. Verify that the Get Features command returns the Power State selected in the Set Features command.

Possible Problems: None.

Test 2.5 – Power State Exit Latency (M) (PCIeC 8.5)

Purpose: To verify that an NVMe system properly documents its exit latency for each Power State.

Reference:

Old Ref : NVMe Specification 8.4
NVM Express Base specification 2.0a : 8.15
NVMe over PCIe Transport Specification 1.0a : 3.6

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: January 23, 2018

Discussion

The controller may support a Power States Exit Latency field, as indicated in the Power State Descriptor table described in the NVMe specification. This field indicates how long it should take, in microseconds, for a controller's power state to transition to another.

Test Setup: See Appendix A

Test Procedure:

1. For each Power State in the Power State Descriptor table with an EXLAT field that is not 0h:
2. Record the ENLAT field of the selected Power State and the EXLAT field of the current Power State
3. Have the NVMe host send a Set Feature Command with Feature Identifier 02h, Power Management, indicating the new Power State
4. Wait the EXLAT of the previous Power State and the ENLAT of the power state being selected.
5. Send a Get Features command with Feature Identifier 02, Power Management.

Observable Results:

1. Verify that the Get Features command returns the Power State selected in the Set Features command.

Possible Problems: None.

Test 2.6 – Relative Read Throughput (M) (PCIeC 8.6)

Purpose: To determine if an NVMe Controller properly supports the Relative Read Throughput rates supported by different power states correctly.

Reference:

Old Ref : NVMe Specification 8.4
NVM Express Base specification 2.0a : 8.15
NVMe over PCIe Transport Specification 1.0a : 3.6

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: January 23, 2018

Discussion

Different Power States have different Relative Read Throughput values that determine the speed at which they can read from the NVMe.

Test Setup: See Appendix A

Test Procedure:

1. Record the Relative Read Throughput for each supported power state.
2. For each power state, record the time it takes to do 100 read operations.

Observable Results:

1. Verify that the times for the 100 reads for a power state is lower than the time for any power state with a higher RRT.

Possible Problems: None.

Test 2.7 – Relative Write Throughput (M) (PCIeC 8.7)

Purpose: To determine if an NVMe Controller properly supports the Relative Write Throughput rates supported by different power states correctly.

Reference:

Old Ref : NVMe Specification 8.4
NVM Express Base specification 2.0a : 8.15
NVMe over PCIe Transport Specification 1.0a : 3.6

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: January 23, 2018

Discussion

Different Power States have different Relative Write Throughput values that determine the speed at which they can write to the NVMe.

Test Setup: See Appendix A

Test Procedure:

1. Record the Relative Write Throughput for each supported power state.
2. For each power state, record the time it takes to do 100 writes.

Observable Results:

1. Verify that the times for the 100 writes for a power state is low than the time for any power state with a higher RWT.

Possible Problems: None.

Test 2.8 – Host Controlled Thermal Management (M) (PCIeC 8.8)

Purpose: To determine if an NVMe Controller properly supports the Host Controlled Thermal Management feature correctly.

Reference:

Old Ref : NVMe Specification 8.4
NVM Express Base Specification 2.0a : 5.27.1.13, 8.15.5

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: November 26, 2018

Discussion

The Host Controlled Thermal Management command can be set within certain boundaries specified by the Minimum Host Thermal Management Temperature field and the Maximum Host Thermal Management field in the Identify Controller Structure, this test ensures that the behavior of a Set Features command with an FID specifying Host Controlled Thermal Management reacts correctly to improper Thermal Management 1 & 2 values.

Test Setup: See Appendix A

Case 1: Basic Operation (M)

Test Procedure:

1. Record the current temperature of the DUT.
2. Perform an Identify Controller Data Structure (CNS=01h) to the DUT. Check the HCTMA (Host Controlled Thermal Management Attribute) bit. If HCTMA is set to 0, this test is not applicable. If HCTMA is set to 1, proceed to the next step.
3. Set the drive to the highest active power state
4. If possible, send a Set Features with FID 10h (Host Controlled Thermal Management), with a Thermal Management Temperature 1 that is below the current device temperature, and a Thermal Management Temperature 2 that is above the current temperature.
5. Record the current power state.
6. Set the drive to the highest active power state
7. If possible, send a Set Features with FID 10h (Host Controlled Thermal Management), with a Thermal Management Temperature 1 that is below the current device temperature, and a Thermal Management Temperature 2 that is above the Thermal Management Temperature 1, but below the current temperature.
8. Record the current power state.

Observable Results:

1. Verify that after each Set Features command completes correctly.
2. For informational purposes display the Power State after each Set Features command.

Case 2: Invalid Field (M)

Test Procedure:

1. Record the current power state.
2. Perform an Identify Controller Data Structure (CNS=01h) to the DUT. Check the HCTMA (Host Controlled Thermal Management Attribute) bit. If HCTMA is set to 0, this test is not applicable. If HCTMA is set to 1, proceed to the next step.
3. Perform a Set Features with FID 10h (Host Controlled Thermal Management), with a Thermal Management Temperature 1 that is 1 degree above the allowed, and a Thermal Management Temperature 2 that is 0xFFFF.

4. Record the current power state.
5. Perform a Set Features with FID 10h (Host Controlled Thermal Management), with a Thermal Management Temperature 1 that is 0xFFFF, and a Thermal Management Temperature 2 that is 1 degree above the Minimum Thermal Management Temperature.
6. Record the current power state.
7. Perform a Set Features with FID 10h (Host Controlled Thermal Management), with a Thermal Management Temperature 1 that is 1 degree above the allowed threshold, and a Thermal Management Temperature 2 that is 0x1.
8. Record the current power state.
9. Perform a Set Features with FID 10h (Host Controlled Thermal Management), with a Thermal Management Temperature 1 that is 0x1, and a Thermal Management Temperature 2 that is 1 degree above the Minimum Thermal Management Temperature.
10. Record the current power state.

Observable Results:

1. Verify that after each Set Features command, an error code of Invalid Field is indicated and the power state has not changed.

Possible Problems: None.

Group 3: System Bus Registers (PCIeC Group 10)

Overview:

This section describes a method for performing conformance verification for NVMe products implementing the System Bus Register.

Notes:

The preliminary draft descriptions for the tests defined in this group are considered complete, and the tests are pending implementation (during which time additional revisions/modifications are likely to occur).

Test 3.1 – PCI Express Capability Registers (M) (PCIeC 10.1)

Purpose: To determine if an NVMe Controller properly implements the PCI Express Capability Registers.

References:

Old Ref : NVMe Specification 2.5
NVMe over PCIe Transport Specification 1.0a : 3.8.5

Resource Requirements:

Tools capable of monitoring and decoding traffic on the NVMe interface.

Last Modification: June 13, 2016

Discussion: NVMe controller using the PCI Express system bus must report their PCI Express capabilities via the PCI Express Capability Registers.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the Host and Controller to bring up the PCIe link, and bring the NVMe Controller to the enabled state.
2. Examine the PCIe bringup sequence for the PXCAP register contents, and record. The PCIe capability structure ID is 0x10.

Observable Results:

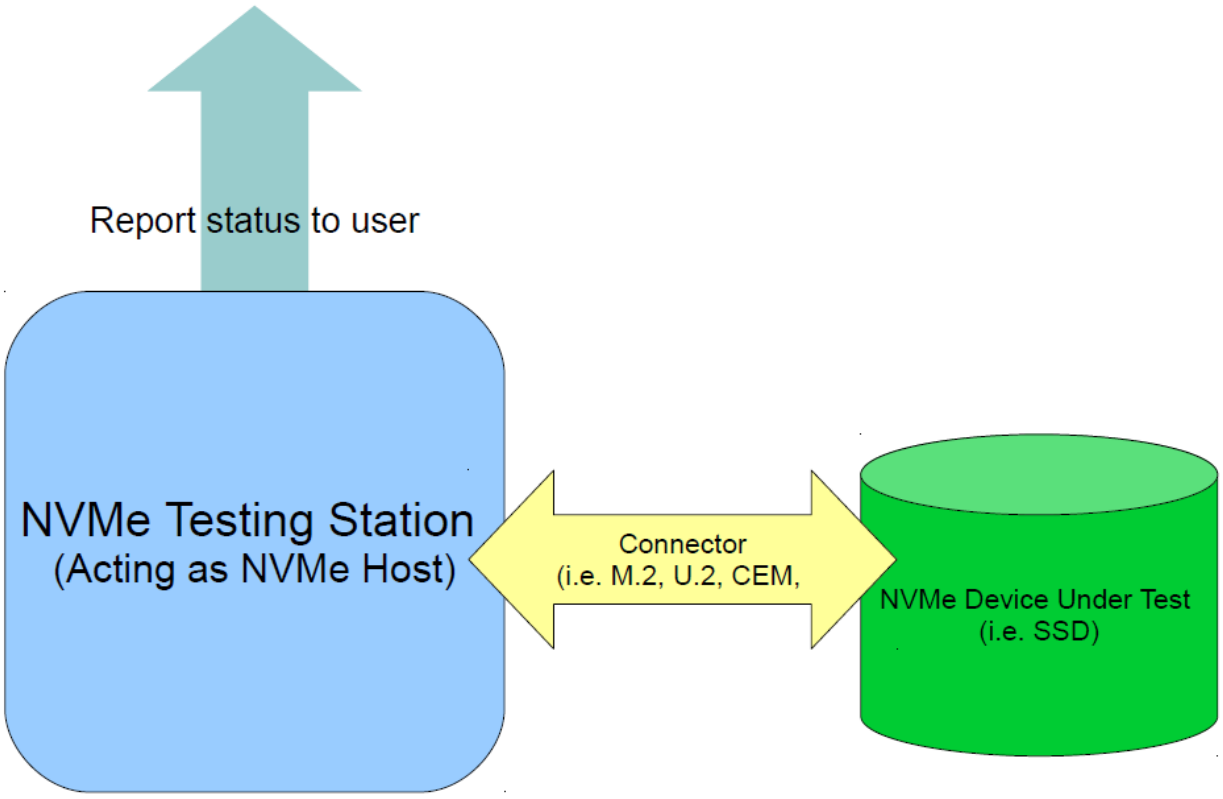
1. Verify that the PCIe Capabilities register (PXCAP) follows the format defined in section 2.5 of the NVMe Specification.
2. Verify that all reserved bits are set to 0.

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:

Test Setup for NVMe Device:



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: TEST TOOLS

The Tests described in this document can be performed using available IOL INTERACT NVMe Test Software available from UNH–IOL.

If using the PC Edition of the IOL INTERACT NVMe Test Software, UNH-IOL recommends using v16.0 or higher of the IOL INTERACT NVMe Test Software. This software is available via https://www.iol.unh.edu/solutions/test-tools/interact.

If using the Teledyne-LeCroy edition of the IOL INTERACT NVMe Test Software, UNH-IOL recommends using v16.0 or higher of the IOL INTERACT NVMe Test Software. This software is available at https://www.iol.unh.edu/solutions/test-tools/interact. This should be used in conjunction with the latest Teledyne-LeCroy software available at: http://teledynelecroy.com

Appendix D: NVMe Integrators List Requirements

Purpose: To provide guidance on what tests are required for NVMe Integrators List Qualification

References:

[1] NVMe Integrators List Policy Document

Resource Requirements:

NVMe Host Platform and Device.

Last Modification: April 2, 2019

Discussion: Each Test defined in this document is defined as being Mandatory (M), FYI, or In Progress (IP). This primary designation is shown in the title of the test case and is understood to apply to PCIe based products. An additional designation is provided if a test is applicable to NVMe-oF products (OF). Tests that are designated as being applicable to NVMe-oF Products are understood to inherit the primary designation of the test (i.e. M, FYI, IP), unless an additional designation is specified. The following examples are provided:

Test 1.1 Example Name (M)– Test is mandatory for all PCIe based products and does not apply to NVMe-oF products.
Test 2.1 Example Name (M, OF)– Test is mandatory for all products, including NVMe-oF products.
Test 3.1 Example Name (M, OF-IP)- Test is mandatory for all PCIe based products, and test is currently On Progress for NVMe-oF products.

NVMe protocol testing is independent of the transport used. Conformance tests described in this document may be performed at any link speed, width, or transport type that the NVMe product under test supports.

If a Test is designated as Mandatory, a product must pass this test in order to qualify for the NVMe Integrators List. For tests that deal with features defined as optional in the NVMe specification, a check is performed at the beginning of the test to determine if the optional feature is supported or not. If the optional feature is not supported the test is marked as 'Not Applicable' and does not impact qualification for the Integrators List.

If a Test is designated as FYI, a device does not need to pass this test in order to qualify for the NVMe Integrators List. Tests designated as FYI may become Mandatory tests in the future.

If a Test is designated as In Progress, a device does not need to pass this test in order to qualify for the NVMe Integrators List. These test cases are still under development. Tests designated as In Progress may become Mandatory tests in the future.

Any Test may have a Case within it with a different designation as the Test itself (i.e. a Mandatory test may include FYI cases). In this case, only the Mandatory Cases are required for NVMe Integrators List qualification.