UNH-IOL
NVMe Test Consortium

Test Suite for NVMe Interoperability
Version 1.1b
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 : September 18, 2014

UNH-IOL NVMe Consortium
121 Technology Dr Suite 2
Durham, NH 03824

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

TABLE OF CONTENTS

MODIFICATION RECORD

2012 May 7 (Version 0.1) Initial Release
David Woolf:

2012 June 21 (Version 0.2)
Raju Mishra:

2012 November 5 (Version 0.3)
David Woolf: Editorial Fixes

2013 April 30 (Version 0.4)
David Woolf: Preparation for plugfest, reorganization of interop tests to be more OS focused.

2013 May 21 (Version 1.0)
David Woolf: Clarification of test steps refined during May 2013 NVMe Plugfest.

2013 September 10 (Version 1.1)
David Woolf: Changes to test 1.3 to allow for use of stressing data patterns and varying transfer sizes. Addition of tests 1.4, 1.5, 1.6.

2013 December 16 (Version 1.1 DRAFT)
David Woolf: Modified Test 1.5 to clarify what OS media will be used, and that the OS install will occur using the UEFI NVMe driver. Modified Tests 1.1 and 1.2 to clarify procedure for identifying and formatting a drive in Windows and Linux operating systems. Updated Appendix A for latest version of vdbench. Added Appendix B. Added Appendix C.

2013 December 19 (Version 1.1 DRAFT)
David Woolf: Modified Appendix A

2013 December 23 (Version 1.1 DRAFT)
David Woolf: Corrected link to vdbench parameter file in Appedix A. Modified the command to start vdbench to include the '-vr' modifier to cause all writes to be validated immediately.

2014 March 11 (Version 1.1)
David Woolf: Corrected link to vdbench parameter file in Appedix A to account for corner case discovered during February 2014 NVMe Plugfest.

2014 March 31 (Version 1.1)
David Woolf: Added notes to the Possible Problems section of Test 1.3, and Appendix A, to account for problems that may arise when testing devices that are less than 512 MB in storage capacity.

2014 April 7 (Version 1.1)
David Woolf: Added information to Appendix C and Test 1.5.

2014 July 10 (Version 1.1)
David Woolf: Added test 1.7

2014 July 14 (Version 1.1b)
David Woolf: Added Appendix D. Added note to all tests to refer to Appendix D if using a non-CEM form factor. Renamed document t o version 1.1b, to match latest NVMe specification and the IL policy to be used for the next plugfest.

2014 August 21 (Version 1.1b)

David Woolf: Edited Appendix A, and test 1.2 and 1.3 to clarify that an NVMe device should be unformatted when performing Test 1.3.

2014 August 25 (Version 1.1b)

David Woolf: Further clarifications to test 1.3 and 1.4 to clarify that an NVMe device should be unformatted when performing these tests.

2014 September 18 (Version 1.1b)

David Woolf: Clarifications to test 1.6 (hotplug) procedure and observable results.

ACKNOWLEDGMENTS

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

David Woolf 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 interoperates with other products designed to the NVM Express Revision 1.1b specification, hereafter referred to as the "NVMe Specification"). Successful completion of these tests provide a reasonable level of confidence that the Device Under Test (DUT) will function properly in many NVMe environments.

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 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:

1. NVM Express Revision 1.1b Specification (July 2, 2014)
2. PCI Express Base Specification Revision 3.0 (November 10, 2010)

Group 1: OS Based Interop Tests

Overview: This section describes a method for performing interop verification for NVMe products using features readily available in the most common operating systems.

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 – Storage Device Identified

Purpose: To verify that an NVMe Host can properly identify an attached NVMe Storage Device.

References:

[1] NVMe Specification

Resource Requirements:

NVMe Host Platform and Device.
If using a non-CEM form factor, see Appendix D

Last Modification: December 16, 2013

Discussion: NVMe Hosts should be able to identify all attached NVMe Devices and display these in the operating system.

Test Setup: All products are powered off. The NVMe device is physically attached to the NVMe Host Platform.

Test Procedure:

1. Power on the NVMe Host and allow the OS to boot and all necessary drivers to be loaded.
    a. In Windows based systems the attached NVMe device should be visible under Control Panel > Administrative Tools > Computer Management > Disk Management.
    b. In Linux based systems the 'lspci' command can be used to list all attached PCI devices, followed by 'ls /dev/nvme*'.

Observable Results:

1. Verify that all NVMe devices are identified by the Host and can be displayed to the user through the operating system.

Possible Problems: Methods for performing this test may vary across different operating systems and drivers.

Test 1.2 – Format Storage Device

Purpose: To verify that an NVMe Host can properly format an attached NVMe Storage Device.

References:

[1] NVMe Specification

Resource Requirements:

NVMe Host Platform and Device.
If using a non-CEM form factor, see Appendix D

Last Modification: August 21, 2014

Discussion: NVMe Hosts should be able to format all attached NVMe Devices and display these in the operating system.

Test Setup: The NVMe device is physically attached to the NVMe Host Platform has been identified by the Host Platform, and is visible to the user.

Test Procedure:

1. Use available OS tools to format the NVMe Device, then write a file to the device. See Appendix B for how to do this in Windows and Linux operating systems.
2. Delete the partition created when the drive was formatted.

Observable Results:

1. Verify that the drive can be partitioned and formatted, a file can be successfully written to the NVMe Device, and the partition can be deleted.

Possible Problems: Methods for performing this test may vary across different operating systems and drivers. Be sure to delete the created partition when the test is complete.

Test 1.3 – Write Read Compare

Purpose: To verify that an NVMe Host can write and read to an attached NVMe Storage Device without error.

References:

[1] NVMe Specification

Resource Requirements:

NVMe Host Platform and Device.
If using a non-CEM form factor, see Appendix D

Last Modification: August 25, 2014

Discussion: NVMe Hosts should be able to write and read files to an attached NVMe Device without error.

Test Setup: The NVMe device is physically attached to the NVMe Host Platform has been identified by the Host Platform, and is visible to the user.

Test Procedure:

1. Remove or delete any formatting on the NVMe device.
2. Use available tools to perform write/read/compare data operations containing a stressing data pattern to the NVMe Device for 5 minutes. The data operations should be of varying transfer sizes. See Appendix A.
3. Shutdown the Host Platform. Power on the Host Platform.
4. Repeat steps 1 and 2, 3 times.
5. Restart the Host Platform.
6. Repeat steps 1 and 4, 3 times.

Observable Results:

1. Verify that for all cases, the write/read/compare operations complete without error to the NVMe Device.
2. Verify that the steps in the test procedure are performed correctly, and that from steps 1-5, there are no errors reported during the performing of the tests.

Possible Problems: Methods for performing this test may vary across different operating systems and drivers. If the DUT has a storage capacity less than 512MB, it will be necessary to modify the threads parameter in the vdbench parameter file. Change the ‘threads=128’ to ‘threads=16’. See Appendix A for more detail on the use of vdbench.

Test 1.4 – Multiple Devices on Bus

Purpose: To verify that an NVMe Host can write and read to an attached NVMe Storage Device without error when multiple NVMe devices are active in the system.

References:

[1] NVMe Specification

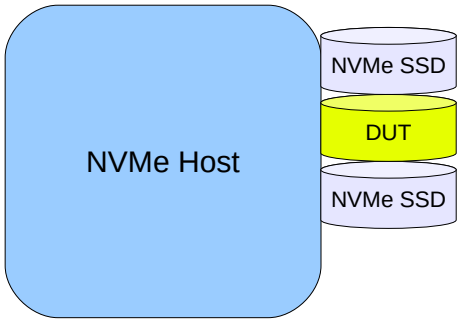
Resource Requirements:

NVMe Host Platform and Device.
If using a non-CEM form factor, see Appendix D

Last Modification: August 25, 2014

Discussion: NVMe Hosts should be able to write and read files to an attached NVMe Device without error.

Test Setup: The NVMe device is physically attached to the NVMe Host Platform and is visible to the user. Additional NVMe device are attached to the NVMe Host in the slots immediately adjacent to the slot containing the device under test.



Test Procedure:

1. Remove or delete any formatting on the NVMe device.
2. Use available tools to perform write/read/compare data operations containing a stressing data pattern to the attached NVMe Device for 5 minutes. The data operations should be of varying transfer sizes. See Appendix A.
3. Shutdown the Host Platform. Power on the Host Platform.
4. Repeat steps 1 and 2, 3 times.
5. Restart the Host Platform.
6. Repeat steps 1 and 4, 3 times.

Observable Results:

1. Verify that for all cases, the write/read/compare operations complete without error to the NVMe Device.
2. Verify that the steps in the test procedure are performed correctly, and that from steps 1-5, there are no errors reported during the performing of the tests.

Possible Problems: Methods for performing this test may vary across different operating systems and drivers.

Test 1.5 – Boot from NVMe Device (Optional)

Purpose: To verify that an NVMe Host can boot from an attached NVMe Storage Device.
References:

[1] NVMe Specification

Resource Requirements:

NVMe Host Platform and Device.
If using a non-CEM form factor, see Appendix D

Last Modification: July 14, 2014

Discussion: NVMe Hosts should be able to boot from an attached NVMe Device without error. At this time, this test is not used to determine qualification of products for the NVMe Integrators List.

Test Setup: The NVMe device is physically attached to the NVMe Host Platform. The Host platform will use the UEFI NVMe Driver to boot from the NVMe device.

Test Procedure: The procedure below describes installing Windows Server 2012 R2 on an NVMe host platform with UEFI support. The procedure will vary for different operating systems. The system used was a **Dell PowerEdge R720 with BIOS Revision 2.2.2**. Other host platforms are acceptable.

1. Download the X64 version of the UEFI shell by going to http://sourceforge.net/apps/mediawiki/tianocore/index.php?title=UEFI_Shell
2. Navigate by clicking the links to ShellBinPkg -> UefiShell/ X64 -> Shell.efi
3. Prepare a USB key to store the UEFI shell by formatting the key as FAT32 and then copy the Shell.efi to the root directory on the key
4. !! also get NvmExpressDxe.x64.efi onto root of the thumb drive !!
5. Power off system (Never hot plug an NVMe device)
6. Install NVMe device.
7. Insert UEFI shell thumb drive in a USB port
8. Power on system
9. Insert Windows installation disc in optical drive.
10. Press function key to start UEFI boot manager (for example on Dell systems this is usually F11)
11. Select “UEFI Boot Menu”
12. Select “Boot From File”
13. Select “[UEFI] Disk connected to front USB…”
14. Select “Shell.efi”
15. Press ‘ESC’ within 5 seconds when prompted.
16. At the command prompt, select the thumb drive by typing “fs0:”
17. Type: “NvmExpressDxe.x64.efi” (you can use the tab key to complete the filename) and press enter
18. Type: “connect” and press enter, the system should respond with several “Connect – Handle [xx] Result Success” messages
19. Restart the shell by typing “exit” to return to the Boot Manager menu.
20. Select “UEFI Boot Menu” then “Boot From File” then “[UEFI] Disk connected to front USB.”
21. Select “Shell.efi”
22. Press ‘ESC’ within 5 seconds when prompted.
23. Type “fs1:” to select the optical drive
24. Type “cd efi” to Change Directory to the EFI directory on the installer disc
25. Type “cd boot” to Change Directory to the boot directory
26. Type “bootx64.efi” to start the installation process, be ready to press any key when prompted to boot from the optical disc
27. Follow the Windows installer prompts to format the NVMe device and complete the install
28. Once the install completes the system reboots
29. Press function key to start UEFI boot manager (for example on Dell systems this is usually F11)

30. Select “UEFI Boot Menu”
31. Select the NVMe device, which may appear as an unknown device.
32. OS will boot

Observable Results:

1. Verify that the operating system booted as expected, the NVMe device was identified properly by the operating system for each power down or reboot cycle.

Possible Problems: Methods for performing this test may vary across different operating systems and drivers. Not all operating systems may have boot support for NVMe. Additionally different hardware platforms may behave differently, with slightly different steps necessary.

Test 1.6 – Hotplug NVMe Device (Optional)

Purpose: To verify that an NVMe Host properly handle hotplugging of an NVMe Storage Device.

References:

[1] NVMe Specification

Resource Requirements:

NVMe Host Platform and Device supporting SFF-8639 form factor.

Last Modification: September 18, 2014

Discussion: NVMe Devices should be able to hotplugged from/to an NVMe Host without error. At this time, this test is not used to determine qualification of products for the NVMe Integrators List. This test is considered optional, and for information only.

Test Setup: At least 2 NVMe devices are physically attached to the NVMe Host Platform and are visible to the user in the OS.

Test Procedure:

1. Power on the NVMe Host and allow the OS to boot and all necessary drivers to be loaded.
2. Ensure that the OS has not loaded from the NVMe device under test, but has loaded from another source.
3. Using the appropriate OS utilities, disable the NVMe device. In Windows, this may be under 'Device Manager'. In Linux, it may be necessary to unmount the NVMe device. Methods may vary across OSes. Other host platforms may have vendor specific methods for performing orderly device removal.
4. Physically remove the device from the NVMe Host.
5. Verify that the removed device is no longer visible in the OS.
6. Verify that the Host is still able to access the remaining NVMe device.
7. Physically reinstall the device to the same slot on the NVMe host. Use any necessary platform specific tools to perform an orderly device addition.
8. Format the both NVMe device installed in the host platform and begin read/write/compare operations as defined in Test 1.3.

Observable Results:

1. Verify that after step 8 in the procedure both NVMe devices are identified by the Host OS, formatted without error, and read/write/compare operations completed without error.

Possible Problems: Methods for disabling the NVMe device may vary across different operating systems.

Test 1.7 – Dual Port Device (Optional)

Purpose: To verify that an NVMe Host properly handles a NVMe Storage Device with Dual Ports.

References:

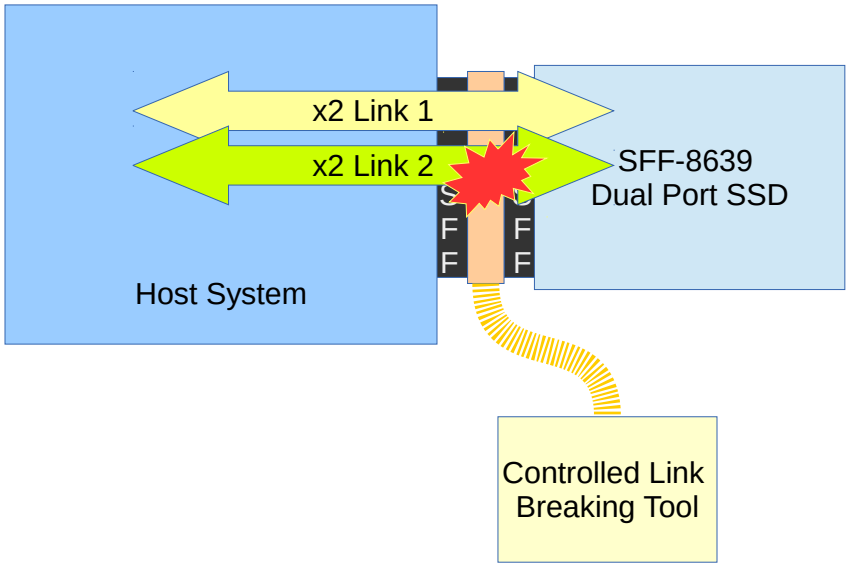
[1] NVMe Specification

Resource Requirements:

NVMe Host Platform and Device.
If using a non-CEM form factor, see Appendix D

Last Modification: July 10, 2014

Discussion: Dual Ported NVMe device provide redundancy in the face of link breakages or port failures. This test only applies to SFF-8639 devices that support dual ports. A controlled method must be used to break and repair links in this test, as shown in the diagram below. In cabled interfaces this may involve disconnecting the cable. In a non cabled interface such as PCIe, specialized hard will be necessary.



Test Setup: The NVMe dual ported device is physically attached to the NVMe Host Platform has been formatted by the Host Platform, and is visible to the user.

Test Procedure:

1. Power on the NVMe Host and allow the OS to boot and all necessary drivers to be loaded.
2. Ensure that the OS has not loaded from the NVMe device under test, but has loaded from another source.
3. Using the appropriate OS utilities determine if the dual port NVMe device appears twice in the OS, or if some method of MPIO is enabled, the NVMe device appears on ly once.
4. Using a controlled method, break one link to the NVMe device, while the other link remained connected and active.
5. Using the appropriate OS utilities determine the following
    a. If the dual port NVMe device appeared twice in the OS, now only 1 instance of the NVMe device is visible in the Host OS.
    b. If some method of MPIO is enabled, the NVMe device still appears in the Host OS.
6. Using a controlled method, reconnect the broken link to the NVMe device, while the other link remains connected and active.

Observable Results:

1. Using the appropriate OS utilities determine the following
    a. If the dual port NVMe device appeared twice in the OS in step 3, there is now again 2 instances of the NVMe device visible in the Host OS.
    b. If some method of MPIO is enabled, the NVMe device still appears in the Host OS.

Possible Problems: Methods for disabling the NVMe device may vary across different operating systems.

Appendix A - Write/Read/Compare Utility for Windows and Linux

Purpose: To define a means of connecting prototype NVMe products for performing the tests outlined in this document.

References:

[1]NVMe Specification

Resource Requirements:

NVMe Host and Device, PCIe conformance channel

Last Modification: August 21, 2014

Discussion: To perform interoperability testing between an NVMe Host Platform and an NVMe Device, a utility capable of writing, reading, and comparing data written to an NVMe Device is necessary. This Appendix will describe the use of the VDBENCH utility to perform this testing. Being a Java based tool, VDBENCH has the benefit of working across multiple operating systems. Other utilities may be used instead of VDBENCH, but their behavior should be known to be similar to that implemented with VDBENCH.

Stressing patterns can be used in the test by using a file with contents that will produce a pattern on the PCIe bus when the file transfer occurs.

Test Setup: The NVMe device is physically attached to the NVMe Host Platform. The NVMe device has not been formatted by the Host Platform, all partitions have been deleted, and is visible to the user.

Test Procedure:

1. Install vdbench 504, following the instructions at http://www.oracle.com/technetwork/server-storage/vdbench-downloads-1901681.html
2. Download the VDBENCH parameter file for your OS:
    a. https://www.iol.unh.edu/services/testing/NVMe/unh_interop_1_1_windows.txt
    b. https://www.iol.unh.edu/services/testing/NVMe/unh_interop_1_1_linux.txt
3. Verify that the attached NVMe Device is visible in OS (disk management in windows or lspci in linux)
4. Edit the unh_interop.txt VDBENCH parameter file so that the 'lun' parameter points to the SSD under test.
5. Run the unh_iol_1_1.txt parameter file using the following command for vdbench: vdbench -f unh_iol_1_1_<linux/windows>.txt –vr.
6. Verify that the vdbench test run completed with no errors. Check /vdbench504/output/summary.html

Observable Results:

1. Verify that the write/read/compare operations complete without error to the NVMe Device.

Possible Problems:

1. Methods for performing this test may vary across different operating systems and drivers.
2. If VDBENCH indicates a Java error, it may be necessary to update the path to Java in vdbench.bat. To do this, use the following steps:
    a. Open vdbench.bat for editing.
    b. Find where the java path is in the bat file, by default it says: set java=java
    c. Find where java is on the system. For example, in Windows 7 64 bit and Windows Servier 2012 systems the path is C:\Program Files (x86)\Java\jre7\bin\java
    d. Copy the new path into the vdbench.bat file: set java="C:\Program Files (x86)\Java\jre7\bin\java". Be sure to use the double quotes.
3. If the DUT has a storage capacity less than 512MB, it will be necessary to modify the threads parameter in the vdbench parameter file. Change the 'threads=128' to 'threads=16'. If this adjustment is not made, the test may not complete, and vdbench may indicate a 'busy' or 'timeout' error.

Appendix B - Formatting Drive

Purpose: To define the procedure for formatting a drive in Windows and Linux Operating Systems.

References:

[1]NVMe Specification

Resource Requirements:

NVMe Host and Device, PCIe conformance channel

Last Modification: December 16, 2013

Discussion: To perform interoperability testing between an NVMe Host Platform and an NVMe Device, a utility capable of writing, reading, and comparing data written to an NVMe Device is necessary. This Appendix will describe the use of the VDBENCH utility to perform this testing. Being a Java based tool, VDBENCH has the benefit of working across multiple operating systems. Other utilities may be used instead of VDBENCH, but their behavior should be known to be similar to that implemented with VDBENCH.

Stressing patterns can be used in the test by using a file with contents that will produce a pattern on the PCIe bus when the file transfer occurs.

Test Setup: The NVMe device is physically attached to the NVMe Host Platform has been formatted by the Host Platform, and is visible to the user.

Windows Procedure:

1. Navigate to Control Panel, Administrative Tools, Computer Management, Disk Management.
2. Right click on the volume to be formatted.

Linux Procedure:

1. sudo fdisk /dev/nvme0n1, then press 'w'
2. sudo mkfs.ext2 /dev/nvme0n1
3. sudo mkdir /mnt/nvme
4. sudo mount /dev/nvme0n1 /mnt/nvme
5. sudo cp unhiol.tst /mnt/nvme
6. sudo ./vdbench -f unh_interop.txt

Note that steps 1 and 2 are only necessary when initially installing the drive. After reboot, only step 4 is necessary

Observable Results:

1. Verify that the write/read/compare operations complete without error to the NVMe Device.

Possible Problems:

Methods for performing this test may vary across different operating systems and drivers.

Appendix C - Write/Read/Compare Utility for UEFI

Purpose: To define a means of connecting prototype NVMe products for performing the tests outlined in this document when using UEFI.

Last Modification: April 3, 2014

Discussion: An NVMe Driver exists for the UEFI boot shell. Below is a procedure for performing a Write/Read/Compare operation from a UEFI Host to an NVMe drive to match the requirements of Test 1.3 in this document..

Test Setup: The NVMe device is physically attached to the NVMe Host Platform.

Test Procedure: The Write/Read/Compare tests over UEFI can be run off of a USB stick in a system that can boot from USB. To prep a USB stick for testing, simply extract the IHV-SCT_Binary (This is used for device testing) and the UEFI-SCT_Binary (This is used for platform testing) as well as the compatible shell version.

Download the X64 version of the UEFI shell by going to
http://sourceforge.net/apps/mediawiki/tianocore/index.php?title=UEFI_Shell
Navigate by clicking the links to ShellBinPkg -> UefiShell/ X64 -> Shell.efi

Later releases of the SCT test software may support newer versions of the shell.
To check on versions of the SCT test software, visit http://www.uefi.org/specs/access which has a list of all releases. Follow the instructions on the site to get access to the new releases.

UEFI Test Guide for IHV-SCT Testing **(**Note**:** the steps below may vary for different host platforms)

1. Load a USB drive with the correct folder of SCT testing and shell
2. Start the system and enter boot manger when prompted
3. Select:
    a. "UEFI Boot Menu"
    b. "Boot From File"
    c. "Cruzer Glide" USB option
    d. "Shellv1.efi"
4. Stop automatic startup
5. EFI might start in an incorrect drive
    a. Type "cd .." until at the root of the drive
    b. Type "ls" and you should see the following:
        1.*******SUBJECT TO CHANGE*******
        2.UEFI-SCT_Binary
        3.Shellv1.efi
        4.Other
        5.SCT
        6.Startup.nsh
    c. If these sections are not seen:
        1.Type map –b and try each of the fsX: locations
        2.Change locations by typing "fsx:"
6. Now that you are in the correct location navigate to "\IHV-SCT_Binary\IHVSct\SctPackageX64\X64"
7. Type "sct –u"
8. Enter "Test Device Configuration"
    a. Type "s1"
    b. It should be obvious which drive is NVMe by looking at the drivers, it doesn't specifically say NVMe but through process of elimination only one should remain
    c. To add the device to the list, type "I XX" where XX is the reference handle (example: C6)
    d. Choose "P" for PCIe device
    e. Choose "Y" for all for default configuration

9. Quit device configuration with “q”
10. Enter “Test Case Management”
    a. Enter the IHV folder and activate the following;
        1.“DevicePathProtocolTest”
        2.“DriverModelTest”
        3.“BootableImageSupportTest”
    b. Hit F9 to begin testing
11. Once complete choose the “Test Report Generator” and save the results
    a. These can later be found in \IHV-SCT_Binary\IHVSct\SctPackageX64\X64\Report

Notes: This method takes around 7 hours to run though the USB. A faster method involves transferring all test files to a hard drive, which is advised as this method only takes ~30min to run.

Observable Results:

1. Verify that the SCT tests completed without error.

Possible Problems:

Appendix D – Using Non-CEM Form Factors

Purpose: To define a means of connecting prototype NVMe products for performing the tests outlined in this document when using non-CEM Form Factors.

Last Modification: July 14, 2014

Discussion: An NVMe Drive may be implemented in a variety of form factors, including CEM, SFF-8639, and M.2 If a host system supporting the needed drive form factor is not available, an adapter may be used to connect the drive and host for the purpose of protocol interoperability testing. Examples of such adapters are provided below for reference only. Adapters with similar functionality and specifications are acceptable.

Serial Cables PCI-AD-X439-01 Adapter