UNH–IOL
NVMe Testing Service

Test Plan for NVMe Conformance
Version 15.0
Target Specification: NVMe 1.4
Technical Document

Last Updated: May 3, 2021

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

2012 June 20 (Version 0.1) Initial Release
Raju Mishra:

2012 July 20 (Version 0.2)
Neeraj Gill: Addition of Groups 4 and 5.

2012 September 27 (Version 0.3)
David Woolf: Editorial Fixes and addition of Group 6 tests covering resets.

2013 February 18 (Version 0.31)
David Woolf: Added note regarding 4k sector sizes to tests 2.2 and 2.3.

2013 April 30 (Version 0.32)
David Woolf: Added note that tests 1.5 and 2.2 cannot be performed.

2013 May 9 (Version 0.34)
David Woolf: Added test 2.6.

2013 May 21 (Version 1.0)
David Woolf: Changed version of test suite to 1.0 after completion of NVMe Plugfest.

2013 October 22 (Version 1.1 DRAFT)
David Woolf: Updates to tests 2.1 and 2.4.

2013 December 16 (Version 1.1 DRAFT)
David Woolf: Updates to tests 1.1, 1.2, 1.3, 1.5, 2.1, 2.2, 2.3, 2.4, 2.5, 4.3, 4.4, 4.5, 4.6, 4.7, and 4.8. Most of the changes involve clarifying or removing observable results that could not be easily observed.

2013 December 19 (Version 1.1)
David Woolf: Added Appendix C with information about potential test tools. Removed references to conformance testing of NVMe Hosts, as this is not currently a requirement for including NVMe Hosts on the NVMe Integrators List.

2014 May 27 (Version 1.1)
David Woolf: Edited procedures to test 1.2 and 4.5 to make them easier to perform.

2014 June 5 (Version 1.1)
David Woolf: Corrected test numbering in Group 5 and Group 6. Clarified that tests 3.1, 3.2, and 4.6 are optional. Corrected reference in test 5.3.

2014 July 7 (Version 1.1)
David Woolf: Added test 6.4.

2014 July 10 (Version 1.1)
David Woolf: Added test 2.7, modified procedure to test 2.1.

2014 July 14 (Version 1.1b)
David Woolf: Change specification references to 1.1b revision of specification.

2014 August 14 (Version 1.1b)
David Woolf: Added Group 7 tests for Reservations.

2014 September 18 (Version 1.1b)

David Woolf: Added notes Group 7 tests for Reservations to indicate that these tests are only applicable to device which support reservations.

2014 September 29 (Version 1.1b)
Mike Bogochow: Updated references and fixed errata.

2014 October 16 (Version 1.1b)
David Woolf: Clarification on Mandatory and Optional tests in Group 2.

2015 April 9 (Version 1.2)
David Woolf: Prepared document for 1.2 revisions.

2015 April 13 (Version 1.2)
David Woolf: Added tests 8.1 and 8.2.

2015 June 24 (Version 1.2.1)
Mike Bogochow: Fixed errata and updated references, procedures, and observable results for all tests in groups 1–6.

2015 November 23 (Version 1.2.1)
Mike Bogochow: Added Group 9 Tests.

2016 March 2 (Version 1.2.2)
Mike Bogochow: Fixed errata, clarified language, and updated procedures and observable results of various existing tests.

2016 March 21 (Version 1.3.0)
Mike Bogochow: Rewrote Groups 7 and 8. Added Test 1.7 - Asynchronous Events. Added Test 1.4 - Case 2: Full Queue Condition. Fixed Purpose wording for most tests.

2016 May 19 (Version 6.0 r01)
David Woolf: Adopted new document numbering scheme, rather than using numbers that are close to the current specification release, test suites will follow a numbering scheme based on the Integrators List version, which is incremented with each plugfest.

2016 May 24 (Version 6.0 r02)
David Woolf: Added tags to titles of FYI tests to help readers determine what tests were mandatory versus FYI. Added Appendix outlining which tests were Mandatory, Optional (dependent on feature support), and FYI. Previously this information was in the NVMe Integrators List Policy Document.

2016 May 26 (Version 6.0 r03)
David Woolf: Added requirement to tests 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 2.3, 2.4, 2.6 that all Reserved fields be checked and that they are set to 0.

2016 June 9 (Version 6.0 r04)
David Woolf: Added tests 1.8,1.9, 2.8, 2.9, 3.4, 3.5, 4.18, 8.3.

2016 June 13 (Version 6.0 r05)
David Woolf: Added tests 2.10, 10.1. Added Case 2 to test 2.9 to address the case where Atomic Boundaries are supported.

2016 June 27 (Version 6.0 r06)
David Woolf: Added new subtests to 1.1, 1.2, 1.3, 1.6, 2.1, 2.3, 2.4, 2.5, 2.6, 2.7. Added checking of CSTS.NSSRO to test 6.4.

2016 June 28 (Version 6.0 r07)

David Woolf: Added new subtests to 1.4. Added Mandatory, Optional, FYI designations to all Tests and Cases within Tests. Edited Appendix D for a clearer description of the meaning of Mandatory, Optional, and FYI test designations and NVMe Integrators List requirements.

2016 July 5 (Version 6.0 r08)

David Woolf: Added Case 9 in Test 1.4, added Case 11 in Test 2.3, added Case 11 in Test 2.4. Added Case 4 and 5 to Test 2.2.

2016 July 7 (Version 6.0 r09)

David Woolf: replaced "Optional" indication with "Mandatory if Supported" in all test titles and Appendix D. Flagged 'Test 1.6 – Format NVM' as "Mandatory if Supported", previously it had been indicated as "Mandatory". Replaced 'optional' with "Mandatory if Supported" in the 'Possible Problems' section of several tests.

2016 July 7 (Version 6.0 r10)

David Woolf: Adjusted procedure of Case 2 of test 2.6. Fixed typos in Test 1.6 Test Cases 2 and 3.

2016 July 14 (Version 6.0 r11)

David Woolf: Adjusted procedure of subtest cases in Tests 2.1, 2.3, 2.4, 2.5, 2.6, 2.7 dealing with Invalid Namespace ID Errors.

2016 July 21 (Version 6.0 r12)

David Woolf: Adjusted procedure of Case 1 in Tests 9.2 to check both the case where number of Namespaces is exceeded and the Namespace size is exceeded. Corrected typos in Test Procedure for Test 2.4 cases 8, 9, 10,11 and Test 2.3 case 11. Adjusted procedure in Test 1.3 Cases 7 and 8 so that enough bytes are read to increment the Data Units Read/Written value. Adjusted procedure in Test 2.5 Case 2, 3, 4, 5 and Test 2,7 case 5.

2016 July 21 (Version 6.0 r13)

David Woolf: Adjusted procedures to Test 1.3 Case 1 and 4 to only check Mandatory values. Test 1.3 Case 5 was updated to check that error log entries properly increment. Clarified Test 1.3 Case 4.

2016 August 30 (Version 6.0)

David Woolf: Final version published to UNH-IOL site ahead of October 2016 NVMe Plugfest #6.

2016 September 26 (Version 6.1 r01)

David Woolf:

- Added indication to several tests Group 7, that some cases are only applicable to Dual Port Devices.
- Updated Test 7.1 Case 2 Observable results steps 3 and 7.
- Updated Test 7.2 Case 3 Procedure step 1.b.vi.
- Separated Test 9.2 Case 1 into 2 separate cases, now Test 9.2 Case 1 addressing having too many Namespaces, and Case 3 addressing having insufficient capacity.
- Modified Test 1.3 Case 4 to accommodate for changes between specs 1.2a and 1.2.1.
- Modified Test 2.3 Case 3 and Test 2.4 Case 3.

January 17, 2017 (Version 7.0 r01)

David Woolf:

- Changed indication of FYI to Mandatory or In Progress for many tests based on results of Plugfest #6.
- Corrected Observable Results for Test 1.3 Case 2 and 3 to expect 'Invalid Log Page' rather than 'Invalid Field in Command'.
- Fixed typo in Test 1.5.

- Updated Test 2.5 Case 5 and Test 2.7 Case 5 per ECN 003 of NVMe Specification v1.2.1. This ECN alters behavior expected in earlier versions. Now the expected behavior is each case is that the device report status of Successful.

January 24, 2017 (Version 7.0 r02)

David Woolf:

- Changed indication of FYI to Mandatory or In Progress for tests 3.4 Case 2, and 6.2 based on NVMe Interop and Compliance Committee direction.

February 9, 2017 (Version 7.0 r03)

David Woolf:

- Added FYI Test 1.4 Case 10.

March 22, 2017 (Version 7.0)

David Woolf:

- Final version published to UNH-IOL site ahead of May 2017 NVMe Plugfest #7.

June 17, 2017 (Version 7.0a)

David Woolf:

- Test requirements relaxed in tests 1.1 Case 4, 2.7 Case 9, 2.8 Case 1, 7.1 Case 2, 7.2, and 9.2 Case 3, due to discoveries during May 2017 plugfest. Test Procedure clarified for Test 1.4 Case 7.

August 14, 2017 (Version 8.0)

David Woolf:

- Test requirements relaxed in tests 1.1 Case 4, 2.7 Case 9, 2.8 Case 1, 7.1 Case 2, 7.2, and 9.2 Case 3, due to discoveries during May 2017 plugfest. Test Procedure clarified for Test 1.4 Case 7.
- Clarification in procedure for test 1.2 Case 4.
- Tests 2.3 and 2.4 Case 4 (NLB>MDTS) modified to be 'Mandatory if Supported', as these tests are only applicable if MDTS is not equal to 0.
- Tests 2.3 and 2.4 Case 7 modified to be 'Mandatory if Supported' ,as these tests are only applicable if NN is not equal to 0xFFFFFFFF.
- Test 1.3 Case 4 modified to be Mandatory if Supported depending on whether MDTS=0 or not. Test modified to expect “Invalid Field in Command” if MDTS conflicts with NUMD/NUMDU/NUMDL.
- Added Test Case 1.3 Case 10.
- Modified Test 2.1 to perform compares using a known sequence rather than the Identify Log Page data to simplify test implementation.
- Clarified Test 2.9 Case 1 that both NABSN and NADSPF need to be set to zero for the test to applicable.
- Clarified Test 2.9 Case 2 that either NABSN and NADSPF need to be set to non-zero values for the test to applicable.
- Modified Test 5.5 to make each condition tested into a separate test case. Most test procedures were not modified and so remain Mandatory, with the exception of case 6 which had the procedure modified and is marked FYI.
- Modified Table in Test 5.5 to clarify which status codes were tested and which were not.
- Modified Appendix D to show abbreviations for test case requirements.
- Modified Test 5.5 Case 5, to indicate that if no features are indicated as ‘Not Changeable’ then the test case does not apply.
- Moved Abbreviations from Appendix D to a new section near the beginning of the document, named ‘Abbreviations’.
- Tests 2.5 Case 5, 2.7 Case 5, 3.4 Case 1, 7.3 Case 1, 1.4 Case 9 had status changed from FYI to Mandatory or Mandatory if Supported.
- Tests 1.1 Case 4, 1.7 Case 1, 2.2 Case 3, 4, 5, 2.7 Case 9, 2.9 Case 2, 7.1 Case 2 and 7.2 had status changed from In Progress to FYI.

September 12, 2017 (Version 8.0a)

David Woolf:

- Updated Appendix C.

October 24, 2017 (Version 8.0b)

David Woolf:

- Updated Appendix D to clarify test requirements designations for NVMeoF products.

November 14, 2017 (Version 9.0 draft)

David Woolf:

1. Updated Target Specification on cover page to be NVMe v1.3
2. Updated Default Test Setup diagram in Appendix A.
3. Updated Test 1.3 Case 5 to check the M bit.
4. Fixed error in Test 2.4 Case 3, where Step 1 was referred to in the Test Procedure rather than Step 2, also corrected the Status Code expected in Observable Result step 2 from 0x0A to 0x80. Also updated test to allow for DUT to report status code 0x81, since the status reported would be dependent on which check is performed first, accounting for NVMe v1.3 ECN 002.
5. Fixed error in Test 2.4 Case 6 where a Read command was indicated instead of a Write command.
6. Fixed error in Test 2.4 Case 6 where Step 1 was referred to in the Test Procedure rather than Step 2.
7. Fixed error in Test 2.4 Case 7 where a Read command was indicated instead of a Write command.
8. Fixed error in Test 2.4 Case 7 where Step 1 was referred to in the Test Procedure rather than Step 2.
9. Fixed error in Test 2.3 Case 2, 3, 4, 5, 6, 7 where a data pattern was written, this step was not necessary.
10. Updated Test 4.18 to allow devices to report support for NVMe v1.3. Added check to ensure that VER value from Identify Controller Data structure matches the value in VS.CAP.
11. Updated Test 2.1 Case 4 to match new requirements in NVMe 1.3 ECN 002, which eliminated the requirement for a DUT to report the lowest numerical value status code first.
12. Updated Test 2.3 Case 4 to match new requirements in NVMe 1.3 ECN 002, which eliminated the requirement for a DUT to report the lowest numerical value status code first.
13. Updated Test 2.3 Case 7 to match new requirements in NVMe 1.3 ECN 002, which eliminated the requirement for a DUT to report the lowest numerical value status code first.
14. Updated Test 2.4 Case 4 to match new requirements in NVMe 1.3 ECN 002, which eliminated the requirement for a DUT to report the lowest numerical value status code first.
15. Updated Test 2.4 Case 7 to match new requirements in NVMe 1.3 ECN 002, which eliminated the requirement for a DUT to report the lowest numerical value status code first.
16. Updated Test 2.7 Case 3 to match new requirements in NVMe 1.3 ECN 002, which eliminated the requirement for a DUT to report the lowest numerical value status code first.
17. Updated Test 2.7 Case 5 to ensure that final READ command returns all 0's.
18. Updated Test 2.1 Case 3 to better match the wording in NVMe 1.3 Figure 32.

Removed steps in Test Procedure and Observable Results for Test 2.4 Cases 2-7 where a Read command was used to double check that a Write command had not caused data to be written, since those Read commands themselves could not be performed successfully, and those cases are also checked in Test 2.3

November 28, 2017 (Version 9.0 draft)

David Woolf:

1. Updated Test 1.2 Cases 1-4 Test Procedures and Observable Results to be more complete.
2. Updated Abbreviations section to stop using the 'Mandatory if Supported' nomenclature.
3. Updated Test 1.3 Case 4 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for MDTS value at the beginning of the test to determine if the test is applicable to the DUT or not.
4. Updated Test 1.6 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for OACS field at the beginning of the test to determine if the test is applicable to the DUT or not.
5. Updated Test 2.1 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for ONCS field at the beginning of the test to determine if the test is applicable to the DUT or not.
6. Updated Test 2.2 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for ONCS field at the beginning of the test to determine if the test is applicable to the DUT or not.

7. Updated Test 2.3 Case 4 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for MDTS value at the beginning of the test to determine if the test is applicable to the DUT or not.
8. Updated Test 2.3 Case 7 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for NN value at the beginning of the test to determine if the test is applicable to the DUT or not.
9. Updated Test 2.3 Case 6 to include a check for NN value at the beginning of the test to determine if the test is applicable to the DUT or not.
10. Updated Observable Results of Test 2.3 Case 6 to check for status code 'Invalid Namespace or Format (0BH).
11. Updated Test 2.4 Case 4 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for MDTS value at the beginning of the test to determine if the test is applicable to the DUT or not.
12. Updated Test 2.4 Case 7 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for NN value at the beginning of the test to determine if the test is applicable to the DUT or not.
13. Updated Test 2.5 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for ONCS field at the beginning of the test to determine if the test is applicable to the DUT or not. Added checks for MDTS and NN values where appropriate.
14. Updated Test 2.7 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for ONCS field at the beginning of the test to determine if the test is applicable to the DUT or not. Added checks for MDTS and NN values where appropriate.
15. Updated Test 2.7 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for NABSN field at the beginning of the test to determine if the test is applicable to the DUT or not.
16. Updated Test 3.1 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for Metadata Size field at the beginning of the test to determine if the test is applicable to the DUT or not.
17. Updated Test 3.2 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for DPC field at the beginning of the test to determine if the test is applicable to the DUT or not.
18. Updated Test 3.4 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check for HMPRE field at the beginning of the test to determine if the test is applicable to the DUT or not.
19. Updated Test 4.13 to be 'Mandatory' instead of 'Mandatory if Supported'.
20. Updated Test 5.5 Case 3 to be 'Mandatory' instead of 'Mandatory if Supported'.
21. Updated Test 6.4 to be 'Mandatory' instead of 'Mandatory if Supported'.
22. Updated all tests in Group 7 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check in all tests of the ONCS field to determine of the controller supports reservations.
23. Updated all tests in Group 9 to be 'Mandatory' instead of 'Mandatory if Supported'. Added a check in all tests of the OACS field to determine of the controller supports Namespace Management.
24. Updated Appendix D to remove all references to 'Mandatory if Supported'.
25. Updated Table in Test 1.2 to contain new Features: Timestamp, Keep Alive Timer, Host Controller Thermal Management, Non-Operational Power State Config.
26. Updated Test 2.4 Cases 8, 9, 10 to remove redundant READ operation in Test Procedure.

December 4, 2017 (Version 9.0 draft)

David Woolf:

1. Updated Test 2.5 Case 5 to include more than 1 READ command to be performed after the Write Uncorrectable command in order to ensure that all LBAs affected by the Write Uncorrectable command are affected.
2. Updated Test 2.7 Case 5 to include more than 1 READ command to be performed after the Write Zeroes command in order to ensure that all LBAs affected by the Write Zeroes command are affected.

December 6, 2017 (Version 9.0 draft)

David Woolf:

1. Updated Test 1.6 Case 3 to include a check of FNA Bit 2, to determine if the test case is applicable or not.
2. Updated Test 1.1 Case 5, to show that CNS reserved value testing should check only FFh.

December 12, 2017 (Version 9.0 draft)

David Woolf:

1. Updated Test 1.3 Case 3 to only check LID 7Fh.
2. Updated Test 1.2 Case 5 to only check SEL 111b.
3. Updated Test 2.2 Case 3, Case 4, and Case 5 to be Mandatory for NVMe Drives.
4. Updated Test 1.4 Case 10 to be Mandatory for NVMe Drives.
5. Updated Test 1.7 Case 1 to be Mandatory for NVMe Drives.
6. Updated Test 2.3 Case 11 to be Mandatory for NVMe Drives.
7. Updated Test 2.4 Case 11 to be Mandatory for NVMe Drives.
8. Updated Test 2.7 Case 9 to be Mandatory for NVMe Drives.
9. Updated Test 2.9 Case 2 to be Mandatory for NVMe Drives.
10. Updated Test 5.5 Case 6 to be Mandatory for NVMe Drives.

December 14, 2017 (Version 9.0 draft)

David Woolf:

1. Added Test 1.1 Case 6.
2. Added Test 1.10 Cases 1-8
3. Added Test 1.11 Cases 1-8
4. Added Test 1.12 Cases 1-5
5. Added Test 1.13 Cases 1-4

December 21, 2017 (Version 9.0 draft)

David Woolf:

1. Updated Observable Results in Test 3.4 Case 2 to eliminate items that can not be observed. Changed the test status to FYI.
2. Updated Observable Results in Test 3.4 Case 3 to eliminate items that can not be observed. Changed the test status to FYI.

January 4, 2018 (Version 9.0 draft)

David Woolf:

1. Updated all references in Group 9 from 8.11 to 8.12 to match NVMe v1.3 specification.

January 22, 2018 (Version 9.0 draft)

Colin Dorsey:

1. Added tests 8.4, 8.5, 8.6, 8.7, 8.8. Updated test procedures for test 8.2

January 23, 2018 (Version 9.0 draft)

David Woolf:

1. Updated following tests from OF-FYI to OF to indicate that they are now mandatory for NVMe-oF products, per ICC decision: 1.1 Cases 1-2, 1.3 Cases 1-2, 1.7 Case 2, 2.5 all cases, 2.6 Case 2, 2.8, 3.3 Cases 1-4, 4.1, 4.2, 4.3, 4.4, 4.6, 4.7, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 5.1.

February 1, 2018 (Version 9.0 draft)

David Woolf:

1. Updated test 4.13 test procedure for clarity.
2. Updated test 3.4 Case 3 test procedure for clarity and completeness.

February 5, 2018 (Version 9.0 draft)

David Woolf:

1. Modified Case 6 to Test 1.1 to cover Namespace Identification Descriptors.
2. Added Test 1.3 Cases 11, 12, 13, 14, 15 to address Telemetry Log Pages.

March 15, 2018 (Version 9.0 draft)

David Woolf:

1. Modified Case 3 of Test 1.2 for clarity.

2. Modified Test 7.1 Case 2 and Test 7.2 Case 3 to include the step of performing a Reservation Acquire command.
3. Modified Test 1.2 Case 4, there was a typo indicated DWord 10 rather than DWord 0.

May 21, 2018 (Version 10.0 draft)

David Woolf:

1. Corrected typo in Test 2.2 Case 2

August 30, 2018 (Version 10.0 Release)

David Woolf:

1. 10.0 Release

November 20, 2018 (Version 11.0 draft)

David Woolf:

1. Modified Test 1.1 Case 4 as previous version of test case was invalid.
2. Renumbered Test 1.1. Case 5 as Test 1.1 Case 13.
3. Added new Test 1.1 Case 5 (FYI).
4. Added new Test 1.1 Case 6 (FYI).
5. Added new Test 1.1 Case 7 (FYI).
6. Added new Test 1.1 Case 8 (FYI).
7. Added new Test 1.1 Case 9 (FYI).
8. Added new Test 1.1 Case 10 (FYI).
9. Added new Test 1.1 Case 11 (FYI).
10. Added new Test 1.1 Case 12 (FYI).
11. Added step to Test 1.2 Case 5 to check ONCS Bit 4 to determine if test is applicable before proceeding with test procedure.
12. Updated Test 1.2 Cases 1-4 to check if FIDs are changeable before proceeding with test procedure.
13. Updated Test 1.3 Case 2 to allow for the case where there are no unsupported LIDs because the DUT supports all Vendor Specific LIDs.
14. Expanded Observable Results for Test 1.3 Case 11 (FYI).
15. Added new Test 1.3 Case 16.
16. Updated Test 1.3 Case 3 to use a different value for the reserved value, now uses 0x6F instead of 0x7F.
17. Fixed typo in Test 1.3 Case 4 where the test procedure referenced steps 2 and 3 when it should reference steps 3 and 4.
18. In Test 1.3 Case 4, the reference to the list of mandatory LIDs in the specification was changed to be a list of the mandatory LIDs, instead of a reference.
19. Updated Test 1.3 Case 7 to perform 1000 Read/Compare operations of LBADS, 2x LBADS, and 4x LBADS to ensure that the DUT is following the requirements on recording Data Units Read in 1000 512 Byte units, rather than just recording the number of Read operations, as some devices have been observed to do. Added a step to record the value for LBADS.
20. Updated Test 1.3 Case 8 to perform 1000 Write operations of LBADS, 2x LBADS, and 4x LBADS to ensure that the DUT is following the requirements on recording Data Units Written in 1000 512 Byte units, rather than just recording the number of Write operations, as some devices have been observed to do. Remove the Write Uncorrectable operation from this test. Added a step to record the value for LBADS.
21. Added new Test 1.3 Case 17.
22. Updated Test 1.7 Case to expect a status of 07h Command Abort Requested if a completion queue entry is posted. If no completion queue entry is posted the test is not applicable.
23. Updated Test 1.9 to clarify that the Set Feature command in Step 2b must be performed with the Save bit set to 1 (SV=1).
24. Updated first 2 steps in Test 1.10 Case 1 to clarify how to ensure that no Device Self Test command is currently in progress.
25. Updated first 2 steps in Test 1.10 Case 2 to clarify how to ensure that no Device Self Test command is currently in progress.

26. Updated first 2 steps in Test 1.10 Case 3 to clarify how to ensure that no Device Self Test command is currently in progress.
27. Updated first 2 steps in Test 1.10 Case 4 to clarify how to ensure that no Device Self Test command is currently in progress.
28. Updated first 2 steps in Test 1.10 Case 5 to clarify how to ensure that no Device Self Test command is currently in progress.
29. Updated first 2 steps in Test 1.10 Case 6 to clarify how to ensure that no Device Self Test command is currently in progress.
30. Updated first 2 steps in Test 1.10 Case 7 to clarify how to ensure that no Device Self Test command is currently in progress.
31. Updated first 2 steps in Test 1.10 Case 8 to clarify how to ensure that no Device Self Test command is currently in progress.
32. Updated first 2 steps in Test 1.11 Case 1 to clarify how to ensure that no Device Self Test command is currently in progress.
33. Updated first 2 steps in Test 1.11 Case 2 to clarify how to ensure that no Device Self Test command is currently in progress.
34. Updated first 2 steps in Test 1.11 Case 3 to clarify how to ensure that no Device Self Test command is currently in progress.
35. Updated first 2 steps in Test 1.11 Case 4 to clarify how to ensure that no Device Self Test command is currently in progress.
36. Updated first 2 steps in Test 1.11 Case 5 to clarify how to ensure that no Device Self Test command is currently in progress.
37. Updated first 2 steps in Test 1.11 Case 6 to clarify how to ensure that no Device Self Test command is currently in progress.
38. Updated first 2 steps in Test 1.11 Case 7 to clarify how to ensure that no Device Self Test command is currently in progress.
39. Updated first 2 steps in Test 1.11 Case 8 to clarify how to ensure that no Device Self Test command is currently in progress.
40. Updated first 2 steps in Test 1.12 Case 1 to clarify how to ensure that no Device Self Test command is currently in progress.
41. Updated first 2 steps in Test 1.12 Case 2 to clarify how to ensure that no Device Self Test command is currently in progress.
42. Updated first 2 steps in Test 1.12 Case 3 to clarify how to ensure that no Device Self Test command is currently in progress.
43. Updated first 2 steps in Test 1.12 Case 4 to clarify how to ensure that no Device Self Test command is currently in progress.
44. Updated first 2 steps in Test 1.12 Case 5 to clarify how to ensure that no Device Self Test command is currently in progress.
45. Updated first 2 steps in Test 1.13 Case 1 to clarify how to ensure that no Device Self Test command is currently in progress.
46. Updated first 2 steps in Test 1.13 Case 2 to clarify how to ensure that no Device Self Test command is currently in progress.
47. Updated first 2 steps in Test 1.13 Case 3 to clarify how to ensure that no Device Self Test command is currently in progress.
48. Updated first 2 steps in Test 1.13 Case 4 to clarify how to ensure that no Device Self Test command is currently in progress.
49. Updated Test 2.7 Case 9 to allow for an error code of 81h Invalid Protection Information, as well as 02h Invalid Field in Command.
50. Updated Test 1.10 Case 4 Observable Results step 1 to be the following: Verify that the Device Self-test command returns status Invalid Namespace or Format in Command.
51. Updated Test 1.11 Case 4 Observable Results step 1 to be the following: Verify that the Device Self-test command returns status Invalid Namespace or Format in Command.
52. Updated Test 1.10 Case 6 test procedure to account for the command completion for the DST command to be sent before the DST operation completes.

53. Updated Test 1.10 Case 7 test procedure to account for the command completion for the DST command to be sent before the DST operation completes.
54. Updated Test 1.10 Case 8 test procedure to account for the command completion for the DST command to be sent before the DST operation completes.
55. Updated Test 1.11 Case 6 test procedure to account for the command completion for the DST command to be sent before the DST operation completes.
56. Updated Test 1.11 Case 7 test procedure to account for the command completion for the DST command to be sent before the DST operation completes.
57. Updated Test 1.11 Case 8 test procedure to account for the command completion for the DST command to be sent before the DST operation completes.
58. Updated test procedure for test 4.12 to delete all existing I/O Submission and Completion queues prior to performing test.
59. Updated test procedure for test 5.5 Case 5 to ensure that when attempting to change a feature value, the value offered is different than the current value.
60. Added step to Test 8.8 Case 1, to check the HCTMA bit before proceeding with the test.
61. Added step to Test 8.8 Case 2, to check the HCTMA bit before proceeding with the test.
62. Added step to Test 7.1 Case 1 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
63. Added step to Test 7.1 Case 2 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
64. Added step to Test 7.2 Case 1 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
65. Added step to Test 7.2 Case 2 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
66. Added step to Test 7.2 Case 3 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
67. Added step to Test 7.2 Case 4 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
68. Added step to Test 7.3 Case 1 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
69. Added step to Test 7.3 Case 2 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
70. Added step to Test 7.4 Case 1 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
71. Added step to Test 7.4 Case 2 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
72. Added step to Test 7.4 Case 3 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
73. Added step to Test 7.5 Case 1 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
74. Added step to Test 7.5 Case 2 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
75. Added step to Test 7.5 Case 3 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
76. Added step to Test 7.5 Case 4 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
77. Added step to Test 7.6 Case 1 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
78. Added step to Test 7.6 Case 2 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
79. Added step to Test 7.6 Case 3 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
80. Added step to Test 7.6 Case 4 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

81. Added step to Test 7.7 Case 1 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
82. Added step to Test 7.7 Case 2 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
83. Added step to Test 7.8 Case 1 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
84. Added step to Test 7.8 Case 2 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
85. Added step to Test 7.8 Case 3 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
86. Added step to Test 7.8 Case 4 to perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.
87. Added step to Test 7.6 Case 3 to ensure that the RTYPE field is set to either WRITE EXCLUSIVE ALL REGISTRANTS or EXCLUSIVE ACCESS ALL REGISTRANTS.
88. Updated Test 7.4 Case 2 to align with the updated use of the IEKEY parameter in NVMe v1.3 specification. Expectation is that when the IEKEY parameter is set to 1 in a Reservation Acquire command, that the DUT will return status of "Invalid Field in Command", and that no reservation is acquired.
89. Added new Test 1.2 Case 6.
90. Added new Test 2.2 Case 6.
91. Updated Test 9.2.1 to check the FNA bit, to see whether the DUT supports Format NVM on a per namespace basis, or if all namespaces within the controller must use the same LBAF.
92. Updated Test 9.2.2 to ensure that a namespace exists before attempting to delete a namespace.
93. New test case for Directives Receive, Test 1.15 Case 1.
94. New test case for Directives Receive, Test 1.15 Case 2.
95. New test case for Directives Send, Test 1.16 Case 1.
96. New test case for Directives Send, Test 1.16 Case 2.
97. New test case for Directives Send, Test 1.16 Case 3.
98. Test 1.1 Case 3 Identify Namespace List change from M, OF-FYI to M, OF
99. Test 1.1 Case 5 Identify to Reserved CNS change from M, OF-IP to M, OF-FYI
100. Test 1.3 Case 3 Get Log Page to Reserved LID change from M, OF-FYI to M, OF
101. Test 1.3 Case 4 Get Log Page with NUMD/MDTS Conflict change from M, OF-FYI to M, OF
102. Test 1.7 Case 4 Asynchronous Masking Event change from M, OF-FYI to M, OF
103. Test 2.2 Case 1 Dataset Management change from M, OF-FYI to M, OF
104. Test 2.2 Case 2 Dataset Management Deallocate change from M, OF-FYI to M, OF
105. Test 2.2 Case 5 Dataset Management Deallocate Correct Range change from M, OF-FYI to M, OF
106. Test 2.3 Case 1 Read change from M, OF-FYI to M, OF
107. Test 2.3 Case 3 Read with NLB Out of Range change from M, OF-FYI to M, OF
108. Test 2.3 Case 6 Read with Invalid NSID change from M, OF-FYI to M, OF
109. Test 2.3 Case 7 Read with Invalid NSID and SLBA Out of Range change from M, OF-FYI to M, OF
110. Test 2.3 Case 8 Read with LR=0, FUA=1 change from M, OF-FYI to M, OF
111. Test 2.3 Case 9 Read with LR=1, FUA=0 change from M, OF-FYI to M, OF
112. Test 2.3 Case 10 Read with LR=1, FUA=1 change from M, OF-FYI to M, OF
113. Test 2.4 Case 1 Write change from M, OF-FYI to M, OF
114. Test 2.4 Case 8 Write with LR=0, FUA=1 change from M, OF-FYI to M, OF
115. Test 2.4 Case 10 Write with LR=1, FUA=1 change from M, OF-FYI to M, OF
116. Test 2.6 Case 1 Flush change from M, OF-FYI to M, OF
117. Test 3.3 Case 5 Power Management Feature change from M, OF-FYI to M, OF
118. Test 4.5 CAP.TO change from M, OF-FYI to M, OF
119. Test 4.8 CAP.MQES change from M, OF-FYI to M, OF
120. Test 5.3 Status Field change from M, OF-FYI to M, OF
121. Test 6.1 Conventional Reset change from IP to FYI
122. Test 6.3 Controller Reset change from M, OF-IP to M, OF-FYI
123. Test 9.2 Case 3 Create Namespace when Insufficient Capacity change from IP to FYI

April 2, 2019 (Version 12.0 draft)

David Woolf:

1. Update Test 1.1 Case 4, to include a step where ***active*** namespaces are determined before sending the Identify to each active namespace.
2. Update Test 1.2 Case 4, Observable Result Step 2, to clarify that "bit 0 of Dword 0" is that of the Get Feature command performed in Procedure Step 3a.
3. Update Test 1.2 Case 4, Observable Results Step 4, to accommodate the special case of the Reservation Notification Mask feature, FID 82h. The response to th Get Feature command in step d of the procedure may be "Invalid Namespace" or "Invalid Field in Command" for the Feature ID 82h only, per 5.21.1.20. For all other FIDs the expected response is "Invalid Namepace"
4. Update Test 1.3 Case 2 to check for proper handling of a vendor-specific log page by checking a single known unsupported vendor-specific LID, rather than trying to check the entire range (C0h-FFh) of vendor-specific LIDs.
5. Update Test 1.3 Case 7 to check for support of Compare command before performing compare command. If compare not supported, the test should be not applicable.
6. Update Test 1.3 Case 17, to have a write operation at the end of the test so that the drive remains useable, since the previous operation was a Write Uncorrectable, which will render the drive unreadable.
7. Updated Observable Results in Test 1.5 for clarity.
8. Update Test 1.9 Step 2b, to ensure that rather than checking each FID, check only the FIDs that are both 'Savable' and 'Changeable'
9. Update Test 1.16 Case 2, as the sub case is missing Observable Results. The expected Observable Results should be identical to Case 1, as although the procedure is different, using a different DTYPE value, the end results is that all the commands should complete successfully if Directives are supported.
10. Update Test 2.6.2 to fix typo in Test Procedure numbering.
11. Update Test 3.1 to include support for Fabrics implementations. This requires flagging this test as 'OF-FYI', and tracking the implementations of the test tools.
12. Test. 4.10 – Modify test to check required and maximum values for IOCQES from Identify Controller Data Structure. Verify that maximum value is greater than or equal to minimum value. Verify that IOCQES value in CC register falls within the specified range. Remove checking of CQES as this is not visible from test application.
13. Test 4.11 - Modify test to check required and maximum values for IOSQES from Identify Controller Data Structure. Verify that maximum value is greater than or equal to minimum value. Verify that IOSQES value in CC register falls within the specified range. Remove checking of SQES as this is not visible from test application.
14. Test 5.2 – Modify test to have only one SQ open at a time, instead of having 2 open SQs. Open a SQ and CQ, run a command through those queues and verify that SQID was used correctly. Close the SQ. Open a new SQ with a new SQID. Associate the new SQ with the previous CQ. Run a command through the SQ and CQ, verify that SQID was used properly.
15. Update Tests 5.4 to include support for Fabrics implementations. This requires flagging this test as 'OF-FYI', and tracking the implementations of the test tools.
16. Update Tests 5.5 to include support for Fabrics implementations. This requires flagging this test as 'OF-FYI', and tracking the implementations of the test tools.
17. Update Tests 5.6 to include support for Fabrics implementations. This requires flagging this test as 'OF-FYI', and tracking the implementations of the test tools.
18. Update typo in reference in Test 5.6 to be 4.6.1.2.3 rather than 4.5.1.2.3.
19. Test 6.4 – Modify test to check that drive is removed from host system after NSSR occurs. Then check to see that drive reappears in the host system after a rescan is performed. Remove the checking of the CSTS.NSSRO register as this is in the domain of the host system and not observable from test application.
20. Update test 9.2 Case 1, to have the procedure ensure that any created namespaces are created with the same format as all existing namespaces on the device, if that is required by the device, as indicated by setting FNA to 1. Remove the step where the entire device is formatted.

21. Updated Test 1.4 Case 9 to reflect changes due to NVMe v1.3 ECN 001.
22. Updated Test 1.4 Case 10 to reflect changes due to NVMe v1.3 ECN 001.
23. Updated Test 2.3 Case 11 to reflect changes due to NVMe v1.3 ECN 001.
24. Updated Test 2.7 Case 9, to check for support of End to End Data Protection before proceeding with the test.
25. Updated tests in Group 7 to be FYI for Fabrics products. All dual-port tests in Group 7 made FYI.
26. Updated Observable Results in Test 1.1. Case 1.
27. Updated Observable Results in Test 1.1. Case 4.
28. Added new FYI Test 1.12 Case 6
29. Added new FYI Test 1.13 Case 5
30. Added new FYI Test 1.17 Cases 1 and 2
31. Added new FYI Test 1.2 Case 7.
32. Added new FYI Test 1.2 Case 8.
33. Added new FYI Test 1.3 Case 19.
34. Added new FYI Test 1.6 Case 8.
35. Added new FYI Test 1.6 Case 9.
36. Added new FYI Test 1.6 Case 10.
37. Added new FYI Test 1.6 Case 11.
38. Added new FYI Test 1.6 Case 12.
39. Added new FYI Test 2.1 Case 7.
40. Added new FYI Test 2.6 Case 3.
41. Added new FYI Test 2.11 Case 1.
42. Added new FYI Test 2.11 Case 2.
43. Added new FYI Test 3.4 Case 4.
44. Added new FYI Test 3.4 Case 5.
45. Updated Test Procedure for IP Test 3.5.
46. Added new FYI Tests 3.6 Cases 1 and 2.
47. Added new FYI Tests 3.7 Case 1.
48. Added new FYI Test 7.1 Case 3.
49. Added new FYI Test 7.1 Case 4.
50. Updated Observable Results in FYI Test 7.5 Case 2.
51. Added new FYI Test 7.9 Case 1.
52. Added new FYI Test 7.9 Case 2.
53. Added new FYI Test 7.9 Case 3.
54. Added new FYI Test 9.1 Case 5.
55. Added new FYI Test 9.2 Case 4.
56. Added check for NS Management Support in Test 1.10 Case 5 and 1.11 Case 5.

March 9, 2020 (Version 13.0)

David Woolf:

1. References to Figures and Tables in the NVMe specification which appeared outside of the references section of each description were removed.
2. Test 1.1 Case 4 status updated from (FYI, OF-FYI) to (M, OF)
3. Test 1.1 Case 5, 6, 8, 9, 10 status updated from (FYI, OF-FYI) to (M, OF -FYI).
4. Test 1.2 Case 1 status updated from (M, OF-FYI) to (M, OF).
5. Test 1.2 Case 6 status updated from (FYI, OF-FYI) to (M, OF - FYI).
6. Test 1.3 Case 5 status updated from (FYI, OF-FYI) to (M, OF-FYI).
7. Test 1.10 Case 1, 2, 3 status updated from (FYI, OF-FYI) to (M, OF-FYI).
8. Test 1.11 Case 1, 2, 3 status updated from (FYI, OF-FYI) to (M, OF-FYI).
9. Test 1.12 Case 2, 3, 4, 5 status updated from (FYI, OF-FYI) to (M, OF-FYI).
10. Test 1.13 Case 2, 3, 4 status updated from (FYI, OF-FYI) to (M, OF-FYI).
11. Test 2.2 Case 6 status updated from (FYI, OF-FYI) to (M, OF).
12. Test 2.3 Case 2 3, 5, 6, 7 status updated from (M, OF-FYI) to (M, OF).
13. Test 2.4 Case 2, 3, 5, 6, 7 status updated from (M, OF-FYI) to (M, OF).
14. Test 6.1 status updated from (FYI) to (M).

15. Tests 8.4, 8.5 status updated from (FYI) to (M).
16. 'Target Specification' for Conformance Test Plan has been updated to be NVMe v1.4.
17. Test 1.2.4, updated Step 2a of Observable Results to check for a status of 'Feature Not Saveable' to be returned instead of 'Invalid field in Command' if the feature is not saveable as indicated by Dword 0 bit 0 of the Completion Entry for the Get Feature request for that feature ID. Removed duplicate steps. Returned to FYI status due to significant test changes.
18. Test 1.3.3, updated test case to allow for different Error Codes being reported depending on the specification version supported.
19. Test 1.3.4, added a statement to end of step 4, which says, "If NUMD cannot be set to a value greater than MDTS, or if MDTS =0, then this test is not applicable".
20. Test 1.3.19, updated test description to indicate Data Units Read should increase by 7x LBADS/512 rather than 14x LBADS/512. Currently the UNH-IOL test implementation correctly checks for 7x, just the description needed to be fixed.
21. Test 1.2 Case 8, updated the 2nd step of the Observable Results to have the second command say "Get Features" rather than "Set Features".
22. Updated Test 1.3 Case 6 to make sure that if a device claims support for NVMe v1.4 or higher, that the Get Log Page - SMART / Health Status – Critical Warning Bit 1 is set to 1 only when a temperature is greater than or equal to an over temperature threshold; or less than or equal to an under temperature threshold. See NVMe revision 1.4 section 5.2 & 5.14
23. Modified Test Case 1.5 (now case 1.5.1) to address modifications to the Abort Command, specifically that if the command to abort was successfully aborted, then a completion queue entry for the aborted command shall be posted to the appropriate Admin or I/O Completion Queue with a status of Command Abort Requested before the completion queue entry for the Abort command is posted to the Admin Completion Queue. See NVMe v1.4 section 5.1.
24. For all cases, 1-12, in Test 1.6, steps were added to record DPS settings at the beginning of the test, then when the test is complete, perform another Format NVM operation to restore the DPS settings recorded at the beginning of the test.
25. Test 1.6 Case 5 and 6, clarified that if the DUT supports all 16 LBAF, then the test is not applicable, as all valid LBAF values are supported, and the host cannot request an unsupported value. Renamed both cases with "Unsupported LBAF" instead of "Invalid LBAF". Added a step to the test procedure to check if all 16 LBAF are supported, and if yes, skip the test.
26. Test 1.6 Case 7, check DPC field for support for End to End Data Protection before starting the test. Observable results depend on the DPC values.
27. Test 1.6 Case 7, in Step 6, procedure clarified to say '…LBAF with *non-zero metadata size* supported by the DUT…'
28. Test 1.6 Case 8, in Step 4, text should say '…LBAF with *non-zero metadata size* supported by the DUT…'
29. Test 1.6 Case 8, updated to allow either PIL=1 or PIL=0, depending on what is supported by the DUT, as indicated in the DPC field. Removed the PIL from the title of the test case.
30. Removed Test 1.6 Case 9 NSID=FFFFFFFFh, no namespaces, SES=000, since it is an invalid test case.
31. Test 1.6 Case 10 NSID=FFFFFFFFh, no attached namespaces, SES=000, renamed to Test 1.6 Case 9 NSID=FFFFFFFFh, no attached namespaces, SES=000. Added an initial step where a check of bit 0 of FNA = 1, is performed. If bit 0 of FNA is not 1, the test is not applicable. Also added steps where the namespace is detached after the READ operation in Step 5. In Step 6, removed sentence saying bit 0 of FNA bit should be 1. In Step 6, the namespace should be reattached.
32. Removed Test 1.6 Case 11 NSID=FFFFFFFFh, no namespaces, SES≠000, since it is an invalid test case.
33. Test 1.6 Case 12 NSID=FFFFFFFFh, no attached namespaces, SES≠000, renamed to Test 1.6 Case 10 NSID=FFFFFFFFh, no attached namespaces, SES≠000. Added an initial step where a check of bit 1 of FNA = 1, is performed. If bit 1 of FNA is not 1, the test is not applicable. Also, added

steps where the namespace is detached after the READ operation in Step 4. In Step 6, removed sentence saying bit 1 of FNA bit should be 1. In Step 6, the namespace should be reattached.

34. Test Case 1.17.1 added that a Get Log Page for the Sanitize status Log Page should be performed after each Sanitize operation, and added an Observable Result, to check that the Sanitize Status Log page is updated on successful completion of the SANITIZE command.
35. Test Case 1.17.2 updated so that the test is still performed if the SANICAP field is non-zero. Instead of skipping the test if the SANICAP field is non-zero, the test is performed for each SANITIZE operations, to check that the DUT responds with 'Invalid Field in Command' for operations indicated as not supported in the SANICAP field.
36. Test 2.1 Case 7, step 3 updated to have PRACT set to 1 instead of set to 0. This is a change to the documentation only, as the UNH-IOL test implementation already does this correctly. Also, in Step 4, the reference to Step 1 should instead be a reference to Step 3.
37. Test 3.6 Case 1 and 2, added a step to the test procedure to ensure that Predictable Latency Mode is enabled with a Set Feature command prior to checking the associated Log Pages.
38. Obsoleted and removed Tests 1.4 Case 9, Test 1.4 Case 10, Test 2.3 Case 11, and Test 2.4 Case 11, since there is no requirement to report PRP Offset Invalid errors. This was discovered when checking for compliance issues with NVMe v1.4 ECN 002.
39. In test 7.1.4, added a check of the CTRATT field to determine if the controller supports 128 bit Host identifiers, if not the test is not applicable.
40. Updated Test 7.5.2 which previously stated that step RES_REL4 should issue a Reservation Release command with an invalid CRKEY value, and IEKEY bit set to 0, and RES_REL3 should also use an invalid CRKEY value but with IEKEY bit set to 1. In doing so, RES_REL4 was written to do the same thing as RES_REL1 (invalid CRKEY, and IEKEY=0). RES_REL3 and RES_REL4 should actually use the correct CRKEY value. Therefore the documented procedure has been changed in steps RES_REL3 and RES_REL4 from "supplying any reservation key other than the current reservation key associated with the host in the CRKEY field" to "supplying the correct reservation key that is associated with the host in the CRKEY field". This correction has already been made in the UNH-IOL test implementation.
41. In test 7.9.3, updated test case to cycle 1000 times, rather than wrapping the counter. The reason for this is that wrapping the counter would take a prohibitively long time.
42. Modified Test 9.1.5 to just check for NSID uniqueness, by checking all controllers in the subsystems for their NSIDs. If all NSIDs are unique, test passes. If not all NSIDs are unique, check that the not-unique NSIDs are in fact shared namespaces. This can be done by writing a pattern to the namespace through one controller, and then checking that that pattern can be read through all other controller that are sharing that namespace.
43. Modifications to test case 1.3.18 for the Persistent Event log. First, support for PEL will be checked. Next, the PEL will be requested via Get Log command. Next an event will be caused that will cause an update to the PEL. The PEL will be requested again to see that the event was logged. To check persistence, a controller level reset is performed. At this time persistence across power cycles will not be checked. After the reset completes, the PEL will be checked again and see that the recorded events were still present in the log.
44. New Test Case, 1.3.20, was added to specifically check Data Units Read count for the READ operation only. This was necessary because Test 1.3.7 includes both Read and Compare operations, but is skipped if the DUT doesn't support Compare, therefore the Data Units Read functionality was not being checked for READ operations. Test 1.3.7 properly checks Data Units Read for the Compare Command but does not check the READ command.
45. New Test Case, 1.5.2, to check proper operation of the Abort command when an abort is requested for an Admin Command (previously only IO commands were checked).
46. In Test 1.4 Case 7, some test items were broken out into separate test cases creating a completely new Test 1.4 Case 9 and Test 1.4 Case 10.

47. In Test 1.4 Case 7, some test items were broken out into separate test cases also creating a completely new Test 1.4 Case 10.
48. New Test 2.12 for Fused Operations with several sub tests.
49. New Test Case 7.1.5 to check proper use of the Controller ID field for a dynamic controller.
50. New test 9.1.6 to check properly handling of NSID FFFFFFFFh when Namespace Management is not supported.
51. New Test 3.8.1 for PMR.
52. New Test 3.8.2 for PMR.
53. New Test 3.9.1 for Get LBA Status.
54. New Test 3.10,1, 2, 3, 4, for Improved Performance Parameters.
55. Modified Test 1.2.7.
56. New Test 1.2.9
57. New Test 1.2.10
58. New Test 3.11.1
59. New Test 3.12.1 and 2
60. New Test 3.13.1 and 2
61. New Test 3.14.1 and 2
62. New Test 1.1.14.
63. New Test 3.15.1 and 2.
64. New Test 1.2.11.
65. New Test 1.4.11.
66. New Test 1.3.22.
67. New Test 1.3.23.
68. Updated Test 1.10 to ensure that that all DST tests properly allow the DST operation to run and complete in the background even after a completion is issued.
69. Updated Test 1.11 to ensure that that all DST tests properly allow the DST operation to run and complete in the background even after a completion is issued.
70. Updated Test 1.12to ensure that that all DST tests properly allow the DST operation to run and complete in the background even after a completion is issued.
71. Updated Test 1.13 to ensure that that all DST tests properly allow the DST operation to run and complete in the background even after a completion is issued.
72. New Observable Results for the CNTRLTYPE field in Test 1.1 Case 2.
73. New Test 3.16.1.
74. New Test 3.17.1.
75. New Test Case 2.1.8
76. New Test Case 2.2.17
77. New Test Case 2.3.11
78. New Test Case 2.4.11
79. New Test Case 2.5.6
80. New Test Case 2.7.10
81. New Test Case 2.11.3
82. New Test Case 1.2.12
83. Updated tests 1.3.11-16 to include status OF-FYI.
84. New Test Cases 1.18.1
85. New Test Cases 1.18.2
86. New Test Cases 1.18.3
87. New Test Case 1.16.4
88. New Test Case 1.16.5
89. New Test Case 1.16.6

90. New Test Case 7.2.5
91. Modifications to Observable Results in test 1.3 Case 11.
92. Modified procedure in Test 1.1. Case 1 to match the current UNH-IOL test implementation.
93. Corrected Test 1.1 Case 4 Observable Results
94. Corrected Test 1.1 Case 6 Observable Results
95. Corrected Test 1.1 Case 7 Observable Results
96. Corrected Test 1.1 Case 12 Observable Results
97. Fixed Typo in Test 6.4 test procedure.
98. Clarification to test procedure in Test 9.2.1, which does not require update to current test implementation.
99. Updated Test 7.1 Case 3 and 4 to check that the Host Identifier Feature is supported as required.
100. Updated Test 7.1 Case 1 and 2 to check that the Host Identifier Feature is supported as required.
101. Fixed typo with wrong CNS value in Test 1.1.1, was CNS=03h, should be CNS-02h for Active Namespace ID list.

July 21, 2020 (Version 14.0)

David Woolf:

1. Updated reference to NVMe-MI specification.
2. Identify Command Tests 1.1.4, 1.1.5 status changed from FYI to Mandatory
3. Get/Set Features Tests 1.2.4, 1.2.5 status changed from FYI to Mandatory
4. Fixed issue with observables in Test 1.2.7
5. Test 1.3.5 status changed from FYI to Mandatory
6. Updatedf Observable results in Test 1.6 Case 6.
7. Device Self Test Tests 1.10.1-7 status changed from FYI to Mandatory
8. Device Self Test Extended Tests 1.11.1-4, 6-8 status changed from FYI to Mandatory
9. Abort Device Self Test Operation Tests 1.12.2-5 status changed from FYI to Mandatory
10. Abort Extended DST 1.13.2-4 Test status changed from FYI to Mandatory
11. Dataset Management Tests 2.2.1-6 status changed from FYI to Mandatory
12. Write Uncorrectable Test 2.5.5 status changed from FYI to Mandatory
13. Write Zeroes Test 2.7.5 status changed from FYI to Mandatory
14. Reset Test 6.1 status changed from FYI to Mandatory
15. Reservations Tests 7.1.2, 7.2.1, 7.2.2, 7.2.3, 7.3.1, 7.5.1, 7.5.2, 7.5.4 status changed from FYI to Mandatory
16. Test 1.1.13 status changed from OF-FYI to OF
17. Tests 1.2.2,3,4,5,6,8 status changed from OF-FYI to OF
18. Tests 1.3.5, 6, 7, 17 status changed from OF-FYI to OF
19. Test 1.6.1 status changed from OF-FYI to OF
20. Test 2.3.4 status changed from OF-FYI to OF
21. Test 2.4.4 status changed from OF-FYI to OF
22. Test 2.6.3 status changed from OF-FYI to OF
23. Test 2.7.1-8 status changed from OF-FYI to OF
24. Test 3.6.2 status changed from OF-FYI to OF
25. Test 3.7.1 status changed from OF-FYI to OF
26. Fixed Typo in Test 3.17 which made it appear that this test applied to devices using PCIe transport. Test does not apply to PCIe transport devices. The test only applies to Fabric transport devices.
27. Test 6.3 status changed from OF-FYI to OF
28. Updated Test 1.1.1 to address NVMe v1.4 ECN 001 requirement that when the THINP bit in the NSFEAT field is set to ‘1’, the controller shall track the number of allocated blocks in the Namespace Utilization field, and when the THINP bit in the NSFEAT field is set to ‘0’, the controller shall report a value in the Namespace Capacity field that is equal to the Namespace Size.
29. Updated Test 1.1.2 to address NVMe v1.4 ECN 001 requirement that the reverse domain name in an NQN that uses the non-UUID format shall not be “org.nvmexpress”.
30. New test case 1.1.15 added to address TP 4027 requirements for UUID List and CNS=17h.

31. Updated Test 1.3.11 to address TP 4063 requirements for the Telemetry Host-Initialed Data Generation Number increment.
32. Updated Test 1.3.12 to address TP 4063 requirements for the Telemetry Host-Initialed Data Generation Number increment.
33. Updated Test Case 1.3.21 to relating to Data Units Written to address NVMe v1.4 ECN 001 requirement that both Write Uncorrectable commands and Write Zeroes commands shall not impact the Data Units Written value.
34. Updated Test 1.4 Case 10 to replace 'CAP.MQES' with 'NCQA'.
35. Fixed typo in 1.11.4 which incorrectly said STC=2h corresponded to Short DST.
36. Clarified correct Abort status type in 1.12.1.
37. Clarified correct Abort status type in 1.12.2
38. Clarified correct Abort status type in 1.12.3.
39. Updated Test 1.12.4 to address NVME v1.3 ECN 006 clarification around terminating DST operations with Format NVM when NSIDs are the same between commands.
40. Clarified correct Abort status type in 1.12.5.
41. New Test Case 1.12.7 to address NVME v1.3 ECN 006 clarification around terminating DST operations with Format NVM when Format NVM has NSID=FFFFFFFFh.
42. New Test Case 1.12.8 to address NVME v1.3 ECN 006 clarification around terminating DST operations with Format NVM when DST and Format NVM both have NSID=FFFFFFFFh.
43. Clarified correct Abort status type in 1.13.1.
44. Clarified correct Abort status type in 1.13.2
45. Clarified correct Abort status type in 1.13.3.
46. Updated Test 1.13.4 to address NVME v1.3 ECN 006 clarification around terminating DST operations with Format NVM when NSIDs are the same between commands.
47. Clarified correct Abort status type in 1.13.5.
48. New Test Case 1.13.6 to address NVME v1.3 ECN 006 clarification around terminating DST operations with Format NVM when Format NVM has NSID=FFFFFFFFh.
49. New Test Case 1.13.7 to address NVME v1.3 ECN 006 clarification around terminating DST operations with Format NVM when DST and Format NVM both have NSID=FFFFFFFFh.
50. New Test case 1.16.7 added to address NVMe v1.3 ECN 006 clarification that Releasing of Stream Resources when a namespace is deleted or formatted was not described completely.
51. New test case 1.17.3 added to address TP4014 proper use of the 'Sanitize Config' Feature when the FID is not savable.
52. New test case 1.17.4 added to address TP4014 proper use of the 'Sanitize Config' Feature when NDI=1 and NODRM=1.
53. New test case 1.17.5 added to address TP4014 proper use of the 'Sanitize Config' Feature when NDI=1 and NODRM=0.
54. New test case 1.17.6 added to address TP4014 proper use of the SPROG field with respect to Sanitize Status Log when Sanitize is in progress.
55. New test case 1.17.7 added to address TP4014 proper use of the SPROG field with respect to Sanitize Status Log when Sanitize is not in progress.
56. New Test case 1.19.1 added to address used of the Security Receive command with SECP=00h.
57. New Test case 1.20.2 added to address NVMe v1.3 ECN 006 clarification that The Security Receive command specified a failure status code that does not exist (Invalid Parameter). The correct error code is Invalid Field in Command.
58. New Test case 1.20.1 added to address NVMe v1.3 ECN 006 clarification that The Security Send command specified a failure status code that does not exist (Invalid Parameter). The correct error code is Invalid Field in Command.
59. New Test case 2.12.5 added to address NVMe v1.3 ECN 006 clarification that commands that request an unsupported fused operation fail. However, the specification did not specify the error. The error is now specified.
60. New Test Case 2.13.1 to check proper handling of Read commands when DULBE set to 0.
61. New Test Case 2.13.2 to check proper handling of Read, Verify, and Compare commands when DULBE set to 1.

62. New test case 2.13.3 added to address NVMe v1.4 ECN 001 requirement that a logical block shall be marked as allocated when it is written with a Write command.
63. New test case 2.13.3 added to address NVMe v1.4 ECN 001 requirement that a logical block shall be marked as allocated when it is written with a Write Uncorrectable command.
64. New test case 2.13.3 added to address NVMe v1.4 ECN 001 requirement that a logical block shall be marked as allocated when it is written with a Write Zeroes command that does not deallocate the logical block
65. Updated Test Procedure in 3.8.1, and made this test case Mandatory based on Technical WG feedback.
66. Updated Test Procedure in 3.8.2, and made this test case Mandatory based on Technical WG feedback.
67. Clarified Test Procedure in Test 3.9.1.
68. Updated all test cases in 3.10 to indicate that the NPWG, NPWD, NPDG, and NPDA parameters are contained in the Identify Namespace Data Structure rather than the Identify Controller Data Structure. The UNH-IOL implementation of this test already handles this correctly.
69. Updated Test Procedure in 3.11.1, and made this test case Mandatory based on Technical WG feedback.
70. New test case 3.12.3 added to address requirements for TP 4050 for Endurance Group Info Enhancements.
71. New test case 3.12.4 added to address requirements for proper values of the Endurance Group Identifier.
72. New test case 3.12.5 added to address requirements for proper values of the Endurance Group Identifier
73. New test case 3.12.6 added to address requirements for proper values of the Endurance Group Identifier
74. New test case 3.12.7 added to address TP 4050 requirements for Endurance Group Critical warnings configuration.
75. New test case 3.12.8 added to address requirements for Endurance Groups not supported.
76. Updated test case 3.13.2 to issue a Get Feature for Read Recovery Levels supported FID, and compare this to the RRLS value in the Identify Controller Data Structure, rather than using the Identify Namespace Data Structure. The UNH-IOL test implementation already handles this correctly.
77. Test cases 5.5.7 status is returned to FYI only to address NVMe v1.3 ECN 006 clarification that Requirements for validation in the LBA Range Type feature were made too strict in a previous ECN; those requirements have been relaxed (“shall” changed to “may”) to improve compatibility with earlier versions of the specification. The Possible Problems section of the test was updated to provide more detail on this condition.
78. New Test case 7.6.5 added to address NVMe v1.3 ECN 006 clarification that Reservation Acquire with preempt action and incorrect CRKEY should produce an error of Reservation Conflict.
79. Updated Test 9.1.5 to use CNS=02h rather than CNS=00h.
80. Minor edits to Test 1.1 Case 1.
81. Minor edits to Test 1.2 Case 4.
82. Minor edits to Test 1.4 Case 9.
83. Minor edits to Test 1.4. Case 10.

May 3, 2021 1(Version 15.0)

David Woolf:

Tests with Status Changes:

1. Test 2.6.3 status updated from (OF-FYI) to (OF).
2. Test 3.6.2 status updated from (OF-FYI) to (OF).
3. Test 9.1.6 status updated from (OF-FYI) to (OF).
4. Test 1.2.4, 7, 8, 9 status updated from (FYI) to (M).
5. 1.3.11, 12, 13, 14, 15, 16, 17, 23 (Get Log) status updated from (FYI) to (M).
6. 1.10.4, 5, 6 (DST) status updated from (FYI) to (M).
7. 1.11.4, 5, 6, 7, 8 (DST) status updated from (FYI) to (M).
8. 1.12.1 (Abort DST) status updated from (FYI) to (M).
9. 1.13.1 (Abort DST) status updated from (FYI) to (M).

10. 1.16.4 (Directives) status updated from (FYI) to (M).
11. 1.17.1, 2 (Sanitize) status updated from (FYI) to (M).
12. 2.1.8 (Compare) status updated from (FYI) to (M).
13. 2.2.7 (DSM) status updated from (FYI) to (M).
14. 2.5.6 (Write Uncorrectable) status updated from (FYI) to (M).
15. 2.6.3 (Flush) status updated from (FYI) to (M).
16. 2.8.1 (Atomicity) status updated from (FYI) to (M).
17. 2.9.1, 2 (AWUN) status updated from (FYI) to (M).
18. 2.11.3 (Verify) status updated from (FYI) to (M).
19. 3.6.2 (IO Determinism) status updated from (FYI) to (M).
20. 3.8.1, 2 (PMR) status updated from (FYI) to (M).
21. 7.8.1, 2, 3, 4(Reservation Types) status updated from (FYI) to (M).
22. 8.6 (APST) status updated from (FYI) to (M).
23. 8.7 (APST) status updated from (FYI) to (M).
24. 8.8.1, 2 (APST) status updated from (FYI) to (M).
25. 9.1.5 (NS Mgmt) status updated from (FYI) to (M).
26. 9.2.3, 4 (NS Mgmt) status updated from (FYI) to (M).

Tests with modifications or New Tests:

1. Added Test 1.2 Case 13 for checking new requirements around which FIDs are namespace specific.
2. Removed unnecessary Observable Results in Test 1.3.11.
3. Fixed Typo in Test 1.3.21 test procedure. UNH-IOL script implementation is already correct.
4. Updated test 1.12.6 to check for support for NVMe v1.4 or higher. Clarified Observable Result that status is checked in the Self-test Result Data Structure, not in the Command Completion.
5. Updated test 1.13.5 to check for support for NVMe v1.4 or higher. Clarified Observable Result that status is checked in the Self-test Result Data Structure, not in the Command Completion.
6. Test 2.11.3 test procedure modified to clarify that all Verify commands in this test case are performed with NSID=FFFFFFFFh, namely step 3 of the procedure should be modified to say "Perform a Verify command of LBADS bytes and NSID=FFFFFFFFh. If the namespace is formatted with protection information, ensure the Verify Command of LBADS bytes with NSID=FFFFFFFFh uses the same protection information as the namespace was formatted with, and PRACT=0."
7. Updated test procedure in Test 2.13.1 to include the Write operation necessary to complete the test. This is already included in the UNH-IOL script implementation of this test.
8. Changed the order of commands in Test 3.6 Case 1.
9. Changed the order of commands in Test 3.6 Case 2.
10. Added Test 3.6 Case 3, Predictable Latency Mode Not Enabled.
11. Clarified Test Procedure in Test 3.7 Case 1.
12. Test 3.12.6 test procedure modified to query ENDGIDMAX, then send a Get Feature with FID=18h with ENDGID = ENDGIDMAX+1. The observable result is unchanged.
13. Clarified procedure in Test 5.2. No changes to Observable Results.
14. Test 5.5.3 added a check for OACS Bit 2=1 to make sure the test is applicable.
15. Clarified Test Procedure in Test 7.9 Case 1, 2, 3.
16. Clarified Test Procedure in Test 7.2 Case 1, 2, 3, 4, 5
17. Added Test 1.2 Case 13 for requirements in TP 4009 Domains and Partitioning around the Get Features Commands for feature values that apply to a namespace and the NSID field is set to FFFFFFFFh.
18. Added Test 1.2 Case 14 for requirements in TP 4009 Domains and Partitioning around the Get Features Commands for feature values that apply to a namespace and the NSID field is set to FFFFFFFFh: If the NSID field is set to FFFFFFFFh for the Get Features command, the controller shall, unless otherwise specified fail abort the command with a status code of Invalid Namespace or Format unless otherwise specified.
19. Added Test 1.2 Case 15 for requirements in TP 4009 Domains and Partitioning around the Set Features Command for feature values that apply to a namespace and the NSID field is set to FFFFFFFFh. If the NSID field is set to FFFFFFFFh for the Set Features command, and the MDS bit is set to '1' in the Identify Controller data structure, the controller shall abort the command with Invalid Field in

Command; or If the NSID field is set to FFFFFFFFh for the Set Features command, and the MDS bit is cleared to '0' in the Identify Controller data structure, unless otherwise specified, the controller shall set the specified feature value for all namespaces attached to the controller processing the command.

20. Added Test 1.3.24 to check requirements around the Domain Identifier field in the Endurance Group Log page as specified in TP 4009.
21. Added new Observable Result to Test 1.1.2 to check proper use of the Command Size Limit fields (DMRL, DMRSL, and DMSL) according to TP 4040.
22. Updated Test Procedure in Test 2.5.5 to check for Command Size Limit support.
23. Added Test 2.5.7 to check for proper uses of Non-MDTS Command Size Limits in the Write Uncorrectable Command per TP 4040.
24. Updated Test Procedure in Test 2.7.5 to check for Command Size Limit support.
25. Added Test 2.7.11 to check for proper uses of Non-MDTS Command Size Limits in the Write Zeroes Command per TP 4040.
26. Added Test 2.2.8 to check that a DUT does not return status Command Size Limit Exceeded when ONCS Bit 2 = 1 and DMRL is exceeded per TP 4040.
27. Added Test 2.2.9 to check that a DUT does not return status Command Size Limit Exceeded when ONCS Bit 2 = 1and DMRSL is exceeded per TP 4040.
28. Added Test 2.2.10 to check that a DUT does not return status Command Size Limit Exceeded ONCS Bit 2 = 1 and DMSL is exceeded per TP 4040.
29. Added Test 2.2.11 to check that a DUT returns status Command Size Limit Exceeded when ONCS Bit 2 = 0 and DMRL is exceeded per TP 4040.
30. Added Test 2.2.12 to check that a DUT returns status Command Size Limit Exceeded when ONCS Bit 2 = 0 and DMRSL is exceeded per TP 4040.
31. Added Test 2.2.13 to check that a DUT returns status Command Size Limit Exceeded when ONCS Bit 2 = 0 and DMSL is exceeded per TP 4040.
32. Added Test 2.11.4 to check that the VSL field is checked when a Verify command is performed per TP 4040.
33. Updated procedure for test 2.2.1, 2, 3, 4, 5, 6, 7 to check for Command support non-MDTS Command Size Limit support per TP 4040.
34. Updated procedure for test 2.5.1, 2, 3, 4, 6 to check for Command support non-MDTS Command Size Limit support per TP 4040.
35. Updated procedure for test 2.7.1, 2, 3, 4, 6, 7, 8, 9, 10 to check for Command support non-MDTS Command Size Limit support per TP 4040.
36. Updated procedure for test 2.11.1, 2, 3 to check for Command support non-MDTS Command Size Limit support per TP 4040.
37. Updated Observable Results for Test 4.3 per TP 4056 requirements.
38. Added Test 1.2.16 to check that the I/O Command Set Profile Feature (19h) is implemented if bit 43 is set to '1' in CAP.CSS per TP 4056.
39. Updated Test Procedure for Test 2.8.1 per TP 4056.
40. Added Test 2.8.2 to check requirements around AWUN, AWUPF, and AWWU per TP 4056.
41. Added Test 1.2.17 to check that the controller shall abort a command that specifies an index (IOCSCI) that corresponds to an I/O Command Set Combination that has a value of 0h with a status of I/O Command Set Combination Rejected per TP 4056.
42. Added Test 1.2.18 to check that the controller shall abort a command that specifies an index (IOCSCI) that corresponds to an I/O Command Set Combination that is not supported with a status of I/O Command Set Combination Rejected per TP 4056.
43. Updated Test Procedure and Observable Results for test 1.1.4 around the NID and NIDT fields when CNS=03h per TP 4056.
44. Added Test 1.1.16 to check for proper use of the CSI field across a variety of CNS values per TP 4056.
45. Added Test 1.1.17 to check that the Identify I/O Command Set data structure (CNS 1Ch) is implemented if bit 43 is set to '1' in CAP.CSS per TP 4056.
46. Added Test 1.1.18 to check that Identify Commands with CNS values of 00h, 11h, and 16h complete with the correct error code when the Namespace is not associated with the NVM command set.
47. Added Test 1.1.19 to check requirements around CNS 05h when I/O Command Set associated with the namespace identified by the NSID field does not support the Identify Namespace data structure specified

by the CSI field, that the controller shall abort the command with a status of Invalid Field in Command per TP 4056.

48. Added Test 1.1.20 to check requirements around CNS 05h with NSID=FFFFFFFFh per TP 4056.
49. Added Test 1.1.21 to check requirements around CNS 06h when the command set indicated in the CSI field is not supported per TP 4056.
50. Added Test 1.1.22 to check requirements around CNS 1Bh when the command set indicated in the CSI field is not supported per TP 4056.
51. Added Test 1.1.23 to check requirements around CNS 1Bh when the Namespace ID is invalid, per TP 4056.
52. Added Test 1.1.24 to check requirements around CNS 1Ch and which I/O Command Set Combination fields should have a value of 0h, per TP 4056.
53. Added Test 9.3.3 to check requirements around Namespace Attach commands when the namespace does not support the same I/O Command Set as the controller per TP 4056.
54. Added Test 9.3.4 to check requirements around Namespace Attach commands when the namespace supports the same I/O Command Set as the controller, but that command set is not enabled by the current I/O Command Set Profile feature, per TP 4056.
55. Added Test 1.6.11 to check new requirements around the Format NVM command. If bit 3 in the FNA field is set to '0' and a Format NVM command has the NSID field set to FFFFFFFFh, then the command shall complete successfully.
56. Added Test 1.6.12 to check new requirements around the Format NVM command. If bit 3 in the FNA field is set to '1' and a Format NVM command has the NSID field set to FFFFFFFFh, then the controller shall abort the command with a status code of Invalid Field In Command.
57. Added new Observable Result and slightly modified procedure to test 1.1.2 to check requirements of the MAXDNA field per TP 4078.
58. Added new Observable Result and slightly modified procedure to test 1.1.2 to check requirements of the MAXCNA field per TP 4078.
59. Added new Test 9.3.5 to check that an attempt to attach a new Namespace when the MAXDNA Namespace attachment limit has been met, is aborted with status of Namespace Attachment Limit Exceeded per TP 4078.
60. Added new Test 9.3.6 to check that an attempt to attach a new Namespace when the MAXCNA Namespace attachment limit has been met, is aborted with status of Namespace Attachment Limit Exceeded per TP 4078.

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
Raju Mishra	UNH InterOperability Laboratory
Neeraj Gill	UNH InterOperability Laboratory
Qian Liu	UNH InterOperability Laboratory
Kasra Dalvand	UNH InterOperability Laboratory
Benjamin Novak	UNH InterOperability Laboratory
Mike Bogochow	UNH InterOperability Laboratory
Jeff Hensel	UNH InterOperability Laboratory
Colin Dorsey	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 Revision 1.2 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 1.4 Specification (June 10, 2019)
2. NVM Express Management Interface Revsion 1.1 Specification (April 29, 2019)

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 NVMeoF products

Group 1: Admin Command Set

Overview:

This section describes a method for performing conformance verification for NVMe products implementing the Admin Command Set.

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 – Identify Command (M, OF)

Purpose: To verify that an NVMe Controller can properly execute an Identify command.

References:

[1] NVMe Specification 5.15, NVMe v1.3 ECN 004a

Resource Requirements:

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

Last Modification: December 1, 2019

Discussion: The Identify command returns a data buffer that describes the NVM subsystem, the controller or the namespace(s). The data structure is 4096 bytes in size. The host indicates as a command parameter whether to return the controller or namespace specific data structure. For the namespace data structure, the definition of the structure is specific to the I/O command set selected for use.

The data structure returned is based on the Controller or Namespace Structure (CNS) field. If there are fewer namespace identifiers or controller identifiers to return for a Namespace List or Controller List, respectively, then the unused portion of the list is zero filled. Controllers that support Namespace Management shall support CNS values of 10h–13h.

The Identify command uses the PRP Entry 1, PRP Entry 2, and Command Dword 10 fields. All other Command specific fields are reserved. A completion queue entry is posted to the Admin Completion Queue when the Identify data structure has been posted to the memory buffer indicated in PRP Entry 1.

Test Setup: See Appendix A.

Case 1: **CNS=00h Identify Namespace Data Structure (M, OF)**

Test Procedure:

1. Retrieve a list of all active NSIDs using and Identify Command to CNS 02h.
2. For each namespace in the NVM subsystem, configure the NVMe Host to issue an Identify command specifying CNS value 00h to each active namespace in order to receive back an Identify Namespace data structure for the specified namespace.

Observable Results:

1. Verify that the requested data structure is posted to the memory buffer indicated in PRP Entry 1, PRP Entry 2, and Command Dword 10, and that a command completion queue entry is posted to the Admin Completion Queue.
2. If the specified namespace ID is inactive, verify that the data structure returned by the controller is zero filled.
3. Verify that all received responses have all Reserved fields set to 0.
4. Verify '*n*' the Number of LBA Formats (NLBAF) set, and that any LBA Format Support descriptors beyond the NLBAF (i.e. LBAF$n+i$) are set to zero.
5. Verify that the NGUID or EUI64 field contains a globally unique namespace identifier, or a Namespace UUID is provided in the Namespace Identification Descriptor. If the controller is not able to provide a globally unique identifier in the NGUID or EUI64 field, then this field shall be cleared to 0h.
6. Verify that if the THINP bit in the NSFEAT field is cleared to '0', that the value reported in the Namespace Capacity field that is equal to the Namespace Size;
7. Verify that if the THINP bit in the NSFEAT field is set to '1', the DUT tracks the number of allocated blocks in the Namespace Utilization (NUSE) field.

Case 2: **CNS=01h Identify Controller Data Structure (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue an Identify command specifying CNS value 01h to the controller in order to receive back an Identify Controller data structure.
2. Repeat the previous step for all controllers in the domain.

Observable Results:

1. Verify that the requested data structure is posted to the memory buffer indicated in PRP Entry 1, PRP Entry 2, and Command Dword 10, and that a command completion queue entry is posted to the Admin Completion Queue.
2. Verify that all received responses have all Reserved fields set to 0.
3. If the DUT claims support for NVMe Version 1.2 or higher, verify that the value reported for Version support in the CAP register (VS Offset 08h), matches the value reported in the Identify Controller Data Structure VER field (83:80) and is non zero.
4. Verify that any power state descriptors not supported (i.e. Power States beyond the number reported in the NPSS value (263) in the Identify Controller Data Structure) are set to 0.
5. Verify that any values that are reported as ASCII strings (specifically Serial Number- SN, Model Number – MN, and Firmware Revision – FR) are left justified and padded with spaces (ASCII 20h character) to the right.
6. Verify that if the DUT claims to support NVMe Specification version 1.4 or later, that the CNTRLTYPE field in the Identify Controller Data Structure is not set to 0h.
7. Verify that if the NQN format does not include a UUID (i.e. it is the first type of NQN format) that the reverse domain name in the NQN field is not "org.nvmexpress".
8. Verify that the Command Size Limit fields (DMRL, DMRSL, and DMSL) are either all set to zero values, or all set to non-zero values.
9. Verify that if FNA bit 3 is set to 1, that FNA Bit 0 and 1 are cleared to 0.
10. Verify that the same MAXDNA value is reported by all I/O Controllers in the Domain.
11. Verify that the same MAXCNA value is less than or equal to the MNAN field.

Case 3: **CNS=02h Namespace List (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue an Identify command specifying CNS value 02h and CDW1.NSID value 0 to the controller in order to receive back a Namespace List containing active namespace IDs.
2. For each namespace ID returned in the Namespace List, configure the host to issue an Identify command specifying CNS value 00h to the controller in order to receive back an Identify Namespace data structure for the specified namespace.

Observable Results:

1. Verify that the requested Namespace List is posted to the memory buffer indicated in PRP Entry 1, PRP Entry 2, and Command Dword 10, and that a command completion queue entry is posted to the Admin Completion Queue.
2. For each Identify command in step 2, verify that the returned data structure is not zero filled since each namespace ID in the Namespace List should be active.
3. Verify that all received responses have all Reserved fields set to 0.

Case 4: **CNS=03h Namespace Identification Descriptor (M, OF)**

Test Procedure:

1. Check bit 43 of CAP.CSS.
2. Check the Identify Controller Data Structure to determine what version of the NVMe specification the product under test claims to support. If the product under test does not support NVMe v1.3 or higher, this test is not applicable.
3. Send an Identify Command with CNS=02h to determine all active namespaces in the NVM Subsystem.

4. For each active namespace in the NVM subsystem, as determined in the previous step, configure the NVMe Host to issue an Identify command specifying CNS value 03h to the controller in order to receive back the Namespace Identification Descriptor data structure for the specified namespace.

Observable Results:

1. Verify that the command returns one or more Namespace Identification Descriptor structures that fit into the 4096 byte Identify payload.
2. Verify that the controller does not return multiple descriptors with the same Namespace Identification Descriptor Type (NIDT).
3. Verify that if NGUID and EUI64 are set to 0 in the Identify Namespace Data Structure, the Namespace Identification Descriptor reports a value of type 3.
4. If Bit 43 of CAP.CSS is set to 1, verify that the DUT included the NID value in each Namespace Identification Descriptor returned and that only one descriptor was returned for each NIDT value.

Case 5: **CNS=10h Namespace ID List (M, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure OACS Bit 3 to determine if the product under test supports Namespace Management. If the product under test does not support Namespace Management, this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 10h and a value of 0h for the Namespace Identifier in CDW1.NSID to the controller in order to receive back the Allocated Namespace ID List.

Observable Results:

1. Verify that the command returns the namespace list of allocated NSIDs in increasing order that are greater than the value specified in the Namespace Identifier in CDW1.NSID in the Identify Command with CNS=10h.

Case 6: **CNS=11h Identify Allocated Namespace Data Structure Allocated NSID (M, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure OACS Bit 3 to determine if the product under test supports Namespace Management. If the product under test does not support Namespace Management, this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 10h and a value of 0h for the Namespace Identifier in CDW1.NSID to the controller in order to receive back the Allocated Namespace ID List.
3. Configure the NVMe Host to issue an Identify command specifying CNS value 11h and a Namespace Identifier in CDW1.NSID that is a known allocated Namespace, if available, based on the response to the Identify Command tow CNS=10h in the previous step. If there is no allocated Namespace this test is not applicable.

Observable Results:

1. Verify that the command returns the Identify Namespace Data Structure for the specified allocated namespace.

Case 7: **CNS=11h Identify Allocated Namespace Data Structure Unallocated NSID (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure OACS Bit 3 to determine if the product under test supports Namespace Management. If the product under test does not support Namespace Management, this test is not applicable.

2. Configure the NVMe Host to issue an Identify command specifying CNS value 10h and a value of 0h for the Namespace Identifier in CDW1.NSID to the controller in order to receive back the Allocated Namespace ID List.
3. Configure the NVMe Host to issue an Identify command specifying CNS value 11h and a Namespace Identifier in CDW1.NSID that is a known unallocated Namespace, if available, based on the response to the Identify Command tow CNS=10h in the previous step. If there is no unallocated Namespace this test is not applicable.

Observable Results:

1. Verify that the command returns a zero filled Identify Namespace Data Structure for the specified unallocated namespace.

Case 8: **CNS=11h Identify Allocated Namespace Data Structure Invalid NSID (M, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure OACS Bit 3 to determine if the product under test supports Namespace Management. If the product under test does not support Namespace Management, this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 10h and a value of 0h for the Namespace Identifier in CDW1.NSID to the controller in order to receive back the Allocated Namespace ID List.
3. Configure the NVMe Host to issue an Identify command specifying CNS value 11h and an invalid Namespace Identifier in CDW1.NSID, if available. If there is no invalid NSID this test is not applicable.

Observable Results:

1. Verify that the command completes with status 'Invalid Namespace or Format'.

Case 9: **CNS=12h Namespace Attached Controller List (M, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure OACS Bit 3 to determine if the product under test supports Namespace Management. If the product under test does not support Namespace Management, this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 10h and a value of 0h for the Namespace Identifier in CDW1.NSID to the controller in order to receive back the Allocated Namespace ID List.
3. Configure the NVMe Host to issue an Identify command specifying CNS value 12h and a valid value for Namespace Identifier in CDW1.NSID and Controller ID in CDW10.CNTID.

Observable Results:

1. Verify that the command completes with status success and returns a list of controller identifier that are attached to the namespace specified in the CDW1.NSID field.

Case 10: **CNS=13h Controller List (M, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure OACS Bit 3 to determine if the product under test supports Namespace Management. If the product under test does not support Namespace Management, this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 13h and a valid value for Controller ID in CDW10.CNTID.

Observable Results:

1. Verify that the command completes with status success and returns a list of controller identifiers in the NVM subsystem that may or may not be attached to a namespace.

Case 11: **CNS=14h Primary Controller Data Structure (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure OACS Bit 7 to determine if the product under test supports Virtualization Enhancements. If the product under test does not support Virtualization Enhancements, this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 14h and a valid value for Controller ID in CDW10.CNTID.

Observable Results:

1. Verify that the command completes with status success and returns the Primary Controller Capabilities Structure.

Case 12: **CNS=15h Secondary Controller List (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure OACS Bit 7 to determine if the product under test supports Virtualization Enhancements. If the product under test does not support Virtualization Enhancements, this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 15h and a valid value for Controller ID in CDW10.CNTID.

Observable Results:

1. Verify that the command completes with status success and returns the Secondary Controller List for Controller Identifiers greater than or equal to what was specified in the CDW10.CNTID field.

Case 13: **Identify to reserved CNS Value (M, OF)**

Test Procedure:

1. For each namespace in the NVM subsystem, configure the NVMe Host to issue an Identify command specifying a CNS of FFh.

Observable Results:

1. Verify that the command returns status ‘Invalid Field in Command’ 02h.

Case 14: **CNS=16h Namespace Granularity List (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure CTRATT Bit 7 to determine if the product under test supports Namespace Granularity. If the product under test does not support Namespace Granularity (Bit 7 = 1) this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 16h and a valid value for Controller ID in CDW10.CNTID.

Observable Results:

1. Verify that the Identify command for CNS=16h completes successfully and includes a properly formatted Namespace Granularity List.
2. Verify that the namespace granularity descriptors with an index greater than the value in the Number of Descriptors field are cleared to 0h.

Case 15: **CNS=17h UUID List (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure CTRATT Bit 9 to determine if the product under test supports UUID List. If the product under test does not support UUID List (Bit 9 = 1) this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 17h.

Observable Results:

1. Verify that the Identify command for CNS=17h completes successfully and returns a properly formatted UUID List.
2. Verify that The UUID List
    a. Contains at least one valid UUID.
    b. The UUID 1 field contains a non-zero value.
    c. The UUID 127 field is cleared to 0h
    d. The last UUID entry is cleared to 0 to indicate the end of the UUID List.

Case 16: **CSI Field (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command for each of the following supported CNS values: 00h, 01h, 02h, 03h, 04h, 10h, 11h, 12h, 13h ,14h, 15h, 16h, 17h, 1Ch.

Observable Results:

1. Verify that for the CNS values which are supported by the DUT that the CDW11.CSI field was set to 0h.

Case 17: **CNS=1Ch I/O Command Set Data Structure (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 43 of CAP.CSS. If this bit is set to 0 then this test is not applicable.
2. Perform an Identify Command for CNS=1Ch.

Observable Results:

1. Verify that the Identify Command completes successfully, and the expected Data Structure is returned.

Case 18: **NVM Command Set Not Supported (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command for CNS=07h to obtain the list of Active Namespace ID's associated with a specific command set.
2. For all namespaces that are not associated with the NVM Command set perform the following commands:
    a. Identify Command with CNS=00h.
    b. Identify Command with CNS=11h.
    c. Identify Command with CNS=16h.

Observable Results:

1. Verify that the Identify Command with CNS=00h completes with status 'Invalid I/O Command Set'.
2. Verify that the Identify Command with CNS=11h completes with status 'Invalid I/O Command Set'.
3. Verify that the Identify Command with CNS=16h completes with status 'Invalid Field in Command'.

Case 19: **CNS=05h (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command for CNS=07h to obtain the list of Active Namespace ID's associated with a specific command set.
2. Perform an Identify Command for CNS=05h to a given namespace using a CSI value that indicates an I/O Command Set which is not supported by the by that namespace.

Observable Results:
1. Verify that the Identify Command with CNS=05h completes with status 'Invalid Field in Command'.

Case 20: **CNS=05h Namespace Management Not Supported (FYI, OF-FYI)**

Test Procedure:
1. Check Bit 3 of the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management, then this test is not applicable.
2. Perform an Identify Command for CNS=05h with NSID=FFFFFFFFh.

Observable Results:
1. Verify that the Identify Command with CNS=05h completes with status 'Invalid Namespace or Format'.

Case 21: **CNS=06h (FYI, OF-FYI)**

Test Procedure:
1. Perform an Identify Command for CNS=07h to obtain the list of Active Namespace ID's associated with a specific command set.
2. Perform an Identify Command for CNS=06h to a given namespace using a CSI value that indicates an I/O Command Set which is not supported by the by that namespace.

Observable Results:
1. Verify that the Identify Command with CNS=06h completes with status 'Invalid Field in Command'.

Case 22: **CNS=1Bh (FYI, OF-FYI)**

Test Procedure:
1. Perform an Identify Command for CNS=07h to obtain the list of Active Namespace ID's associated with a specific command set.
2. Perform an Identify Command for CNS=1Bh to a given namespace using a CSI value that indicates an I/O Command Set which is not supported by the by that namespace.

Observable Results:
1. Verify that the Identify Command with CNS=1Bh completes with status 'Invalid Field in Command'.

Case 23: **CNS=1Bh, Invalid NSID (FYI, OF-FYI)**

Test Procedure:
1. Perform an Identify Command for CNS=07h to obtain the list of Active Namespace ID's associated with a specific command set.
2. Perform an Identify Command for CNS=1Bh to an invalid namespace using a CSI value that indicates an I/O Command Set which is supported by the by that namespace.

Observable Results:
1. Verify that the Identify Command with CNS=1Bh completes with status 'Invalid Namespace or Format'.

Case 24: **CNS=1Ch (FYI, OF-FYI)**

Test Procedure:

1. Check bit 43 of the CAP.CSS. If this bit is set to 0 this test is not applicable.
2. Perform an Identify Command for CNS=1Ch using a CSI value that indicates an I/O Command Set which is supported by the by that namespace.

Observable Results:

1. Verify that the DUT returns the I/O Command Set Data Structure and that the first I/O Command Set Combination 'N' field that is set to 0, (N from 1 to 511) is the last one with a non-zero value, and all subsequent I/O Command Set Combinations have a value of 0h.

Possible Problems: None.

Test 1.2 – Set/Get Features Commands (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly execute a Get Features command. Also to verify that the NVMe Controller properly sets feature values after the NVMe Host issues a Set Features command.

References:

[1] NVMe Specification 5.13, 5.21, NVMe v1.3 ECN 002, NVMe v1.3 ECN 004a

Resource Requirements:

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

Last Modification: April 2, 2019

Discussion: The Get Features command retrieves the attributes of the Feature indicated. The Get Features command uses the PRP Entry 1, PRP Entry 2, and Command Dword 10 fields. All other command specific fields are reserved.

The Set Features command specifies the attributes of the Feature indicated. The Set Features command uses the PRP Entry 1, PRP Entry 2, Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15 fields. All other command specific fields are reserved.

The desired feature is specified in the Feature Identifier (FID) field of CDW10. Valid Feature Identifiers are described in Table 1 and Table 2.

Table 1 – Feature Identifiers

Feature Identifier	Optional/ Mandatory	Feature
00h	N/A	Reserved
01h	Mandatory	Arbitration
02h	Mandatory	Power Management
03h	Optional	LBA Range Type
04h	Mandatory	Temperature Threshold
05h	Mandatory	Error Recovery
06h	Optional	Volatile Write Cache
07h	Mandatory	Number of Queues
08h	Mandatory	Interrupt Coalescing
09h	Mandatory	Interrupt Vector Configuration
0Ah	Mandatory	Write Atomicity Normal
0Bh	Mandatory	Asynchronous Event Configuration
0Ch	Optional	Autonomous Power State Transition
0Dh	Optional	Host Memory Buffer
0Eh	Optional	Timestamp
0Fh	Optional	Keep Alive Timer
1.2.1.1.1.1 110h	1.2.1.1.1.2 Optional	1.2.1.1.1.3 Host Controller Thermal Management
1.2.1.1.1.4 111h	1.2.1.1.1.5 Optional	1.2.1.1.1.6 Non-Operational Power State Config.
1.2.1.1.1.7 112h – 77h	1.2.1.1.1.8 NN/A	1.2.1.1.1.9 Reserved
78h – 7Fh	N/A	Refer to NVMe-MI Specification

80h – BFh	N/A	Command Set Specific (Reserved)
C0h – FFh	N/A	Vendor Specific

Table 2 – NVM Command Set Specific Feature Identifiers

Feature Identifier	Optional/ Mandatory	Feature
80h	Optional	Software Progress Marker
81h	Optional	Host Identifier
82h	Optional	Reservation Notification Mask
83h	Optional	Reservation Persistence
84h – BFh	N/A	Reserved

A completion queue entry is posted to the Admin Completion Queue when the controller has completed returning any attributes associated with the Feature. Depending on the Feature Identifier, Dword 0 of the completion queue entry may contain feature information

Test Setup: See Appendix A.

Case 2: **SEL = 000b (M, OF)**

Test Procedure:

1. For each of the features described in Table 1 and Table 2, perform the following:
    a. Configure the NVMe Host to issue a Get Features command with SEL field set to 000b indicating the specified FID value in CDW10 to the controller.
    b. Configure the NVMe Host to issue a Set Features command indicating the specified FID value in CDW10 setting valid values for each feature. If any feature is indicated as not changeable by the controller, testing that FID is not applicable. If all FIDs are indicated as not changeable the test is not applicable..
    c. Configure the NVMe Host to issue a second Get Features command with SEL field set to 000b indicating the specified FID value in CDW10 to the controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify that the controller returns the Feature Information for each FID as specified in of the NVMe Specification for each Get Features command issued by the NVMe Host.
3. Verify that the Feature Information returned in the second Get Features command matches the values which were set by the NVMe Host using the Set Features command, if that feature is changeable.
4. Verify that all received responses have all Reserved fields set to 0.

Case 3: **SEL = 001b (M, OF)**

Test Procedure:

1. Check the Bit 4 of the ONCS field to determine of the DUT supports setting the Save field in a Set Features command to a non-zero value and the Select field of the Get Features command to a non-zero value. If Bit 4 is set to 0 this test is not applicable.
2. Configure the NVMe Host to issue a Set Features command indicating the specified FID value in CDW10 setting valid values for each feature. If any feature is indicated as not changeable by the controller, testing that FID is not applicable. If all FIDs are indicated as not changeable the test is not applicable.
3. For each of the features described in Table 1 and Table 2, perform the following:
    a. Configure the NVMe Host to issue a Get Features command with SEL field set to 000b indicating the specified FID value in CDW10 to the controller.
    b. Configure the NVMe Host to issue a Get Features command with SEL field set to 001b indicating the specified FID value in CDW10 to the controller.

c. Configure the NVMe Host to issue a Set Features command indicating the specified FID value in CDW10 changing the feature values for the feature to the controller, if that feature is changeable.
d. Configure the NVMe Host to issue a Get Features command with SEL field set to 000b indicating the specified FID value in CDW10 to the controller.
e. Configure the NVMe Host to issue a Get Features command with SEL field set to 001b indicating the specified FID value in CDW10 to the controller

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify that the controller returns the Feature Information for each FID as specified in section 5.12 of the NVMe Specification for each Get Features command issued by the NVMe Host.
3. Verify that features values returned in Step b and d have changed.
4. Verify that feature values returned in step c and e have not changed.
5. Verify that all received responses have all Reserved fields set to 0.

Case 4: **SEL = 010b (M, OF)**

Test Procedure:

1. Check the Bit 4 of the ONCS field to determine of the DUT supports setting the Save field in a Set Features command to a non-zero value and the Select field of the Get Features command to a non-zero value. If Bit 4 is set to 0 this test is not applicable.
2. Configure the NVMe Host to issue a Set Features command indicating the specified FID value in CDW10 setting valid values for each feature. If any feature is indicated as not changeable by the controller, testing that FID is not applicable. If all FIDs are indicated as not changeable the test is not applicable.
3. For each of the features described in Table 1 and Table 2, perform ensure that the Feature is supported as saveable by the DUT. If the DUT does not support any features as saveable, this Test Case is not applicable. If the DUT does support certain features as saveable, perform the following:
    a. Configure the NVMe Host to issue a Get Features command with SEL set to 000b indicating the specified FID value in CDW10 to the controller.
    b. Configure the NVMe Host to issue a Set Features command with the SV bit set to 1 indicating the specified FID value in CDW10 setting changed values for the feature to the controller, for all changeable values.
    c. Configure the NVMe Host to issue a second Get Features command with SEL=010b indicating the specified FID value in CDW10 to the controller.
    d. Perform a Controller reset.
    e. Configure the NVMe Host to issue a second Get Features command with SEL=010b indicating the specified FID value in CDW10 to the controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify that the controller returns the Feature Information for each FID as specified in section 5.12 of the NVMe Specification for each Get Features command issued by the NVMe Host.
3. Verify that the Feature Information returned in the second Get Features command matches the values which were set by the NVMe Host using the Set Features command, *and were saved across the reset event.*
4. Verify that all received responses have all Reserved fields set to 0.

Case 5: **SEL = 011b (M, OF)**

Test Procedure:

1. Check the Bit 4 of the ONCS field to determine of the DUT supports setting the Save field in a Set Features command to a non-zero value and the Select field of the Get Features command to a non-zero value. If Bit 4 is set to 0 this test is not applicable.
2. Configure the NVMe Host to issue a Set Features command indicating the specified FID value in CDW10 setting valid values for each feature. If any feature is indicated as not changeable by the controller, testing that FID is not applicable. If all FIDs are indicated as not changeable the test is not applicable.
3. For each of the features described in Table 1 and Table 2, perform the following:
    a. Configure the NVMe Host to issue a Get Features command with SEL=011b indicating the specified FID value in CDW10 to the controller. Record the value of Dword 0 bits 0, 1, and 2 returned in the Completion Entry.
    b. Perform a Set Feature command with the SV field set to 1 for the specified FID.
    c. Perform a Get Feature command with the SEL field set to 000b, with NSID set to an active Namespace.
    d. Perform a valid Get Feature command with SEL set to 000b, followed by a valid Set Feature command with a new value for the specified FID.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify that in step 3b,
    a. If bit 0 of Dword 0 of the completion entry for the Get Feature Command sent in step 3a, is set to 0 (i.e. the feature is not saveable), verify that the controller returns "Feature Identifier Not Saveable (0Dh) to the Set Feature Command in step 3b.
    b. If bit 0 of of Dword 0 of the completion entry for the Get Feature Command sent in step 3a, is set to 1(i.e. the feature is saveable), verify that the Set Feature Command in step 3b completes successfully.
3. Verify that in step 3c,
    a. If bit 1 of Dword 0 of the completion entry is set to 0 (i.e. the Feature Identifier is not namespace specific), the DUT returns the controller specific value for that FID and the command completes successfully.
    b. If bit 1 of Dword 0 of the completion entry is set to 1 (i.e. the Feature Identifier is namespace specific), the DUT returns the namespace specific value for that FID and the command completes successfully.
4. Verify that in step 3d,
    a. If bit 2 of Dword 0 of the completion entry is set to 0, the controller returns Feature not Changeable for the Set Features command.
    b. If bit 2 of Dword 0 of the completion entry is set to 1, the command completes successfully.

Case 6: **SEL = Reserved Value (M, OF)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure ONCS Bit 4 to determine if the product under test supports the Save field. If the product under test does not support the Save field, this test is not applicable.
2. For each of the features described in Table 1 and Table 2, perform a Get Features command with the SEL field set to 111b.

Observable Results:

1. Verify that after the completion of each command, the controller returns Error Status 02h, Invalid Field in command.

Case 7: **SEL = 011b Attempt to Change value indicated as Not Changeable (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Get Feature command to the NVMe Controller for each supported Feature ID with SEL set to 011b to determine which features are changeable. If Dword 0 bit 2 of the completion entry of the Get Features command is set to '1', then the Feature Identifier is changeable. If Dword 0 bit 2 of the completion entry of the Get Features command is cleared to '0', then the Feature Identifier is not changeable.
2. For each Feature indicated as Not Changeable, issue a Set Feature Command for that feature offering a value identical to the current value.

Observable Results:

1. Verify that after the completion of each command, the controller returns status of either "Success" or "Feature not Changeable".

Case 8: **NSID of FFFFFFFFh to Namespace Specific Feature (M , OF-FYI)**

Test Procedure:

1. Check the specification version supported by the DUT. If the DUT supports v1.4 or higher this test is applicable, otherwise this test is not applicable.
2. Configure the NVMe Host to issue a Get Feature command to the NVMe Controller for each supported Feature ID with SEL set to 011b
3. For each Feature ID indicated as Namespace Specific (i.e. Dword 0 bit 1 of the completion entry of the Get Features command is set to '1') perform a Get Feature with SEL=000b and NSID =FFFFFFFFh.
4. For each Feature ID indicated as Namespace Specific (i.e. Dword 0 bit 1 of the completion entry of the Get Features command is set to '1') perform a Get Feature with SEL=000b and NSID =0h.
5. For each Feature ID indicated as Namespace Specific (i.e. Dword 0 bit 1 of the completion entry of the Get Features command is set to '1') perform a set Feature with NSID =FFFFFFFFh.
6. For each Feature ID indicated as Namespace Specific (i.e. Dword 0 bit 1 of the completion entry of the Get Features command is set to '1') perform a set Feature with NSID =0h.

Observable Results:

1. This test checks implementation of NVMe v1.3 ECN 002. DUT's claiming to support NVMe v1.3 or earlier may implement the expected behavior. DUT's claiming to support NVMe v1.4 or later must implement the expected behavior.
2. For DUT's supported NVMe v1.4 or later, verify that the Get Feature Commands with NSID of FFFFFFFFh or 0h complete with status code of Invalid Namespace or Format.
3. For DUT's supported NVMe v1.4 or later, verify that the Set Feature Commands with NSID of FFFFFFFFh complete with status Success.
4. For DUT's supported NVMe v1.4 or later, verify that the Set Feature Commands with NSID of 0h complete with status code of Invalid Namespace or Format.

Case 9: **VWC Feature (M, OF)**

Test Procedure:

1. Check the VWC field of the Identify Controller Data Structure.
2. Perform a Get Feature with SEL=000b for the VWC Feature 06h.
3. Perform a Set Feature for the VWC Feature 06h.

Observable Results:

1. If a volatile write cache is not present, then a Set Features command specifying the Volatile Write Cache feature identifier shall fail with Invalid Field in Command status, otherwise the Set Feature command should complete with status success.
2. If a volatile write cache is not present, then a Get Features specifying the Volatile Write Cache feature identifier should fail with Invalid Field in Command status, otherwise the Get Feature command should complete with status success.

Case 10: **FID 03h, NSID=FFFFFFFFh (M, OF-FYI)**

Test Procedure:

1. Perform a Set Feature with FID 03h and NSID=FFFFFFFFh. If FID=03h Is not supported this test is not applicable.

Observable Results:

1. Verify that the Set Feature command is aborted with status 'Invalid Field in Command'.

Case 11: **Controller Feature Values, NSID=0h or FFFFFFFFh (FYI, OF-FYI)**

Test Procedure:

1. For each changeable FID that applies to the controller, perform a Set Feature with a new value and NSID=FFFFFFFFh, followed by a Get Feature with NSID=FFFFFFFFh.
2. For each changeable FID that applies to the controller, perform a Set Feature with a new value (unique from the previous step) and NSID=0h, followed by a Get Feature with NSID=0h.
3. For each changeable FID that applies to the controller, perform a Set Feature with a new value (unique from the previous step) and NSID=FFFFFFFFh, followed by a Get Feature with NSID=0h.
4. For each changeable FID that applies to the controller, perform a Set Feature with a new value (unique from the previous step) and NSID=0h, followed by a Get Feature with NSID=FFFFFFFFh.

Observable Results:

1. Verify that each Set Feature and Get Feature command completed successfully and that each Get Feature command reported the value as set in the previous Set Feature command.

Case 12: **Multiple Set Features Commands for FID 03h (FYI, OF-FYI)**

Test Procedure:

1. Perform a Set Feature with FID 03h for a specific set of values in the LBA Range Type Data Structure Entry. If FID=03h Is not supported this test is not applicable.
2. Perform a Get Operation for FID 03h.
3. Repeat steps 1 and 2 10 times, with new values in the LBA Range Type Data Structure Entry each time.

Observable Results:

1. Verify in all cases that the Get Feature command returned the values from the most recent Set Feature command.

Case 13: **Timestamp FID 0Eh (FYI, OF-FYI)**

Test Procedure:

1. Check the Identify Controller Data Structure ONCS field Bit 6 to determine if the DUT supports the Timestamp feature. If the DUT does not support the Timestamp feature, then this test is not applicable.
2. Perform a Set Feature with FID 0Eh to set a timestamp value in the DUT, start a timer on the testing station.
3. Perform a Get Feature operation with FID=0Eh to request to current timestamp value.
4. Perform a Controller Level Reset.
5. Perform a Get Feature operation with FID=0Eh to request to current timestamp value.

Observable Results:

1. Verify in all cases that the Set/Get Feature operations for FID=0Eh completed successfully.
2. If the Synch bit (returned in response to the Get Feature command) is set to 0, indicating that the controller does not stop counting, verify that the timestamp returned after the controller level reset matches the timer running on the testing station.
3. If the Timestamp Origin field is set to 000b verify that the timestamp shows the time since the Controller Level Reset.

4. If the Timestamp Origin field is set to 001b, verify that the timestamp reflects the value that was initialized in the Set Features command.

Case 14: **Get Feature Namespace Specific FID Valid NSID (FYI, OF-FYI)**

Test Procedure:

1. Check that the DUT claims to support NVMe v1.4 or higher. If the DUT does not support NVMe v1.4 or higher, then this test is not applicable.
2. Perform a Get Feature operation to a valid NSID with FID=03h, LBA Range Type, which is a namespace specifc Feature.
3. Perform a Get Feature operation to a valid NSID with FID=05h, Error Recovery, which is a namespace specifc Feature.
4. Perform a Get Feature operation to a valid NSID with FID=82h, Error Recovery, which is a namespace specifc Feature.
5. Perform a Get Feature operation to a valid NSID with FID=83h, Error Recovery, which is a namespace specifc Feature.
6. Perform a Get Feature operation to a valid NSID with FID=84h, Error Recovery, which is a namespace specifc Feature.
7. If any of the Features are not supported, then proceed to the next step in the procedure.

Observable Results:

1. Verify that for each supported Feature ID the Completion Queue Entry for the Get Feature command has Dword 0 bit 1 set to '1' to indicate that the feature is namespace specific.

Case 15: **Get Feature Namespace Specific FID NSID=FFFFFFFFh (FYI, OF-FYI)**

Test Procedure:

1. Check that the DUT claims to support NVMe v1.4 or higher. If the DUT does not support NVMe v1.4 or higher, then this test is not applicable.
2. Perform a Get Feature operation to NSID=FFFFFFFFh with FID=03h, LBA Range Type, which is a namespace specifc Feature.
3. Perform a Get Feature operation to NSID=FFFFFFFFh with FID=05h, Error Recovery, which is a namespace specifc Feature.
4. Perform a Get Feature operation to NSID=FFFFFFFFh with FID=82h, Error Recovery, which is a namespace specifc Feature.
5. Perform a Get Feature operation to NSID=FFFFFFFFh with FID=83h, Error Recovery, which is a namespace specifc Feature.
6. Perform a Get Feature operation to NSID=FFFFFFFFh FID=84h, Error Recovery, which is a namespace specifc Feature.
7. If any of the Features are not supported, then proceed to the next step in the procedure.

Observable Results:

1. Verify that if Feature ID 03h, 82h, or 83h is supported, the associated Get Feature command fails with status Invalid Field In Command.
2. Verify that for every other supported Feature ID the command fails with status Invalid Namespace or Format.

Case 16: **Set Feature Namespace Specific FID NSID=FFFFFFFFh (FYI, OF-FYI)**

Test Procedure:

1. Check that the DUT claims to support NVMe v1.4 or higher. If the DUT does not support NVMe v1.4 or higher, then this test is not applicable.
2. Check the MDS bit in the Identify Controller Data Structure.
3. Perform a Set Feature operation to NSID=FFFFFFFFh with FID=03h, LBA Range Type, which is a namespace specifc Feature.

4. Perform a Set Feature operation to NSID=FFFFFFFFh with FID=05h, Error Recovery, which is a namespace specifc Feature.
5. Perform a Set Feature operation to NSID=FFFFFFFFh with FID=82h, Error Recovery, which is a namespace specifc Feature.
6. Perform a Set Feature operation to NSID=FFFFFFFFh with FID=83h, Error Recovery, which is a namespace specifc Feature.
7. Perform a Set Feature operation to NSID=FFFFFFFFh FID=84h, Error Recovery, which is a namespace specifc Feature.
8. If any of the Features are not supported, then proceed to the next step in the procedure.
9. If MDS=0, perform a Get Feature command to each supported FID attempted previously in the test procedure.

Observable Results:
1. Verify that if MDS=0, that the DUT set the specified feature value for all namespaces attached to the controller processing the command if the feature is supported.
2. Verify that if MDS=1, that the Set Feature command for supported Feature ID fails with status Invalid Field in Command.

Case 17: **I/O Command Set Profile Feature (FYI, OF-FYI)**

Test Procedure:
1. Check bit 43 in CAP.CSS. If this bit is set to 0 this test is not applicable.
2. Perform a Get Feature operation with FID=19h, I/O Command Set Profile Feature.

Observable Results:
1. Verify that the Get Feature command completes successfully.

Case 18: **I/O Command Set Combination 0h (FYI, OF-FYI)**

Test Procedure:
1. Check bit 43 in CAP.CSS. If this bit is set to 0 this test is not applicable.
2. Perform a Set Feature operation with FID=19h, I/O Command Set Profile Feature, and include an IOCSCI value which corresponds to an I/O Command Set Combination with a value of 0h.

Observable Results:
1. Verify that the Set Feature command completes with status ‘I/O Command Set Combination Rejected’.

Case 19: **I/O Command Set Combination Not Supported (FYI, OF-FYI)**

Test Procedure:
1. Check bit 43 in CAP.CSS. If this bit is set to 0 this test is not applicable.
2. Perform a Set Feature operation with FID=19h, I/O Command Set Profile Feature, and include an IOCSCI value which corresponds to an I/O Command Set Combination which is not supported by the DUT.

Observable Results:
1. Verify that the Set Feature command completes with status ‘I/O Command Set Combination Rejected’.

Possible Problems: None.

Test 1.3 – Get Log Page Command (M, OF)

Purpose: To verify that an NVMe Controller can properly execute a Get Log Page command.

References:

[1] NVMe Specification 5.14, NVMe v1.3 TP 4007a and TP 4030

Resource Requirements:

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

Last Modification: July 15, 2019

Discussion: The Get Log Page command returns a data buffer that contains the log page requested. The Get Log Page command uses the PRP Entry 1, PRP Entry 2, and Command Dword 10 fields. All other command specific fields are reserved.

The desired log page is specified in the Log Page Identifier (LID) field of CDW10. Valid Log Identifiers are described in Table 3 and Table 4.

Table 3 – Log Page Identifiers

Log Identifier	Optional/ Mandatory	Log Page
00h	N/A	Reserved
01h	Mandatory	Error Information
02h	Mandatory	SMART / Health Information
03h	Mandatory	Firmware Slot Information
04h	Optional	Changed Namespace List
05h	Optional	Command Effects Log
06h	Optional	Device Self Test
07h	Optional	Telemetry Host-Initiated
08h	Optional	Telemetry Controller Initiated
09h – 6Fh	N/A	Reserved
70h		Discovery
71h-7Fh		Reserved for NVMe over Fabrics
80h – BFh	N/A	I/O Command Set Specific
C0h – FFh	N/A	Vendor Specific

Table 4 – NVM Command Set Specific Log Page Identifiers

Log Identifier	Optional/ Mandatory	Log Page
80h	Optional	Reservation Notification
81h	Optional	Sanitize Status
82h – BFh	N/A	Reserved

A completion queue entry is posted to the Admin Completion Queue when the log has been posted to the memory buffer indicated in PRP Entry 1.

Test Setup: See Appendix A.

Case 1: **Supported LIDs (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Get Log Page command specifying each of the Mandatory Log Page Identifiers described in Table 3 and 4 above, to the NVMe Controller.

Observable Results:
1. Verify that a completion queue entry is posted to the Admin Completion Queue when the log has been posted to the memory buffer indicated in PRP Entry 1.
2. In each case verify that the information returned in the Log Page information is formatted according to the respective descriptions for the log pages in section 5.14 of the NVMe Specification.
3. Verify that all received responses have all Reserved fields set to 0.

Case 2: **Unsupported Vendor Specific LIDs (M, OF)**

Test Procedure:
1. Configure the NVMe Host to issue a Get Log Page command specifying a single log identifier (LID) from the vendor specific LID range (C0h-FFh) that is not supported by the NVMe Controller. It may be necessary to query product owner for an unsupported LID from the vendor specific range, or consult product data sheet. If the DUT supports all Vendor Specific LIDs, then this test is not applicable.

Observable Results:
1. Verify that each command completes with an error code of Invalid Log Page 09h.
2. Verify that all received responses have all Reserved fields set to 0.

Case 3: **Reserved LIDs (M, OF)**

Test Procedure:
1. Configure the NVMe Host to issue a Get Log Page command specifying log identifiers of 00h, and 6Fh.

Observable Results:
1. Verify that each command completes with an error code of Invalid Log Page 09h. If the DUT claims support for NVMe specification version 1.3 or earlier, they DUT may also return an error code of 'Invalid Field in Command' 02h.
2. Verify that all received responses have all Reserved fields set to 0.

Case 4: **NUMD/MDTS Conflict (M, OF)**

Test Procedure:
1. Check the MDTS value reported by the DUT in the Identify Controller Data Structure. If MDTS is set to 0 (i.e. unlimited) then this test is not applicable.
2. Check the NVME Specification version supported by DUT using the value reported in the Identify Controller Data Structure VER field (83:80). If the DUT supports v1.2.1 or higher go to step 3. If the DUT supports v1.2a or lower (or VER is set to 0, go to step 4).
3. Set the NUMDU and NUMDL values to be greater than the MDTS value provided by the DUT in the Identify Controller Data Structure. Configure the NVMe Host to issue a Get Log Page command specifying each of the Mandatory Log Page Identifiers (01h – Error Information, 02h – SMAR/Health Information, 03h – Firmware Slot Information).
4. Set the NUMD values to be greater than the MDTS value provided by the DUT in the Identify Controller Data Structure. Configure the NVMe Host to issue a Get Log Page command specifying each of the Mandatory Log Page Identifiers. If NUMD cannot be set to a value greater than MDTS, or if MDTS =0 (i.e. unlimited) then this test is not applicable.

Observable Results:
1. Verify that NVMe Controller returns "Invalid Field in Command Error" in response to the received Get Log Page command.

Case 5: **Get Error Information after Error (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Get Log Page command to LID 01h, Error Information, and record the number of Error log entries.
2. Configure the NVMe Host to issue an Identify Command with a Reserved CNS value.
3. Check that the M bit is set in the Status Field of the completion queue entry associated with the Identify Command. If the M bit is not set to 1, this test is not applicable.
4. Configure the NVMe Host to issue a Get Log Page command to LID 01h, Error Information, and record the number of Error log entries.
5. Configure the NVMe Host to issue an Identify Command with a Reserved CNS value.
6. Check that the M bit is set in the Status Field of the completion queue entry associated with the Identify Command. If the M bit is not set to 1, this test is not applicable.
7. Configure the NVMe Host to issue a Get Log Page command to LID 01h, Error Information, and record the number of Error log entries.

Observable Results:

1. If the M bit is set, verify that NVMe Controller reported an error condition in the Error Information log, and the number of Error Log Entries is incremented by one or more. The reported error condition may or may not be directly related to the Identify Command with Reserved CNS value.

Case 6: **SMART Temperature Threshold (M, OF)**

Test Procedure:

1. Check the NVMe version (VER field) supported by the DUT.
2. Configure the NVMe Host to issue a Set Feature command to set the Critical Temperature Threshold below the current temperature of the DUT.
3. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h).

Observable Results:

1. Verify that NVMe Controller reported the Critical Temperature warning in the SMART/Health Information Log.
2. If the DUT claims support for NVMe v1.4 or higher, verify that the Get Log Page - SMART / Health Status – Critical Warning Bit 1 is set to 1 only when a temperature is greater than or equal to an over temperature threshold; or less than or equal to an under temperature threshold.

Case 7: **Data Units Read Count – Compare (M, OF)**

Test Procedure:

1. Check ONCS bit 0 to determine if the DUT supports the Compare Command. If the Compare Command is not supported, this test is not applicable.
2. Configure the NVMe Host to issue a Identify Namespace Data Structure Command CNS=00 and record the value for LBA Data Size (LBADS) for LBA Format 0.
3. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value.
4. Perform 1000 NVMe READ command of LBADS bytes.
5. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 1).
6. Perform 1000 NVMe Compare command of LBADS bytes.
7. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 1). .
8. Perform 1000 NVMe READ command of 2x LBADS bytes.

9. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 2).
10. Perform 1000 NVMe Compare command of 2x LBADS bytes.
11. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 2).
12. Perform 1000 NVMe READ command of 4x LBADS bytes.
13. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 4).
14. Perform 1000 NVMe Compare command of 4x LBADS bytes.
15. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 4). .

Observable Results:

1. Verify that NVMe Controller reported the Data Units Read value that increased by 14x (LBADS/512) in accord with the NVMe READ and Compare commands that were performed.

Case 8: **Data Units Written (M, OF-FYI)**

Test Procedure:

1. Configure the NVMe Host to issue a Identify Namespace Data Structure Command CNS=00 and record the value for LBA Data Size (LBADS) for LBA Format 0.
2. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value.
3. Perform 1000 NVMe Write commands of LBADS bytes..
4. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Written value should increase by 1).
5. Perform 1000 NVMe Write commands of 2x LBADS bytes..
6. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value, which is expected to increase by 2x LBADS/512. (i.e. if LBADS is 512, then the Data Units Written value should increase by 2).
7. Perform 1000 NVMe Write command of 4x LBADS bytes..
8. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value, which is expected to increase by 4x LBADS/512. (i.e. if LBADS is 512, then the Data Units Written value should increase by 4).

Observable Results:

1. Verify that NVMe Controller reported the Data Units Written value that increased by 7x (LBADS/512) in accord with the NVMe Write Commands that were performed.

Case 9: **Power Cycle Count (IP)**

Test Procedure:

1. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h) and record the Power Cycles value.
2. Power Cycle the DUT.
3. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h) and record the Power Cycles value.

Observable Results:

1. Verify that NVMe Controller reported the Power Cycles value, and that the value incremented properly after performed a power cycle.

Case 10: **NUMD Greater than Log Page Conflict (FYI)**

Test Procedure:

1. Check the NVME Specification version supported by DUT using the value reported in the Identify Controller Data Structure VER field (83:80). If the DUT supports v1.2.1 or higher go to step 2. If the DUT supports v1.2a or lower (or VER is set to 0, go to step 3).
2. Set the NUMDU and NUMDL values to be greater than the Log Page Size required for each of the Mandatory Log Page Identifiers.
3. Set the NUMD values to be greater than the Log Page Size required for each of the Mandatory Log Page Identifiers.

Observable Results:

1. Verify that NVMe Controller sends the requested Log Page, and that the size of the Log Page is defined by NUMD or NUMDU/NUMDL. Verify that valid data exists only between bytes 0 to the Log Page Size. Verify that bytes beyond MDTS up to NUMD or NUMDU/NUMDL are filled with undefined data.

Case 11: **Telemetry Host Initiated Valid Offset Create=1 (M, OF-FYI)**

Test Procedure:

1. Check the Bit 3 of the LPA field Identify Controller Data Structure to determine of the DUT supports Telemetry Host-Initiated and Telemetry Controller–Initiated log pages. If Bit 3 is set to 0 this test is not applicable.
2. Configure the NVMe Host to issue a Get Log Page command specifying Telemetry Host-Initiated Log Page to the NVMe Controller with a Log Page Offset Lower value that is a multiple of 512 bytes, and the Create Telemetry Host-Initiated Data bit is set to 1.
3. Check the Telemetry Host-Initiated Data Generation Number. If the Generation number is less than FFh, repeat the previous step until the Generation Number increments to FFh. When the Generation Number reaches FFh, perform the previous step one more time.

Observable Results:

1. Verify that NVMe Controller sends the requested Log Page following the Telemetry Host-Initiated Log Page including the Telemetry Host-Initiated Data Block 1. Note that the Telemetry contents are vendor defined.
2. Verify that the Telemetry Host-Initiated Data Area 2 Last Block field is greater than or equal to the Telemetry Host-Initiated Data Area 1 Last Block field
3. Verify that the Telemetry Host-Initiated Data Area 3 Last Block field is greater than or equal to the Telemetry Host-Initiated Data Area 2 Last Block field.
4. If present, verify that the IEEE OUI Identifier field is a valid IEEE/RAC assigned identifier that is registered at http://standards.ieee.org/develop/regauth/oui/public.html.
5. Verify that the Telemetry Host-Initialed Data Generation Number increments for each Get Log Page command received with Create=1, and that when the Generation Number reaches FFh, it rolls over to 0h on the next returned Log Page.

Case 12: **Telemetry Host Initiated Valid Offset Create=0 (M, OF-FYI)**

Test Procedure:

1. Check the Bit 3 of the LPA field Identify Controller Data Structure to determine of the DUT supports Telemetry Host-Initiated and Telemetry Controller–Initiated log pages. If Bit 3 is set to 0 this test is not applicable.

2. Configure the NVMe Host to issue a Get Log Page command specifying Telemetry Host-Initiated Log Page to the NVMe Controller with a Log Page Offset Lower value that is a multiple of 512 bytes, and the Create Telemetry Host-Initiated Data bit is set to 1.
3. Configure the NVMe Host to issue a Get Log Page command specifying Telemetry Host-Initiated Log Page to the NVMe Controller with a Log Page Offset Lower value that is a multiple of 512 bytes, and the Create Telemetry Host-Initiated Data bit is set to 0.
4. Configure the NVMe Host to issue a Get Log Page command specifying Telemetry Host-Initiated Log Page to the NVMe Controller with a Log Page Offset Lower value that is a multiple of 512 bytes, and the Create Telemetry Host-Initiated Data bit is set to 1.

Observable Results:

1. Verify that in steps 2, 3, and 4 the NVMe Controller sends the requested Log Page following the Telemetry Host-Initiated Log Page format defined in the NVMe specification.
2. Verify that the Log Page delivered by the NVMe Controller in Step 3 was not updated from the Log Page delivered by the Controller in Step 2, and that the Telemetry Host-Initialed Data Generation Number does not increments.

Case 13: **Telemetry Host Initiated Invalid Offset (M, OF-FYI)**

Test Procedure:

1. Check the Bit 3 of the LPA field Identify Controller Data Structure to determine of the DUT supports Telemetry Host-Initiated and Telemetry Controller–Initiated log pages. If Bit 3 is set to 0 this test is not applicable.
2. Configure the NVMe Host to issue a Get Log Page command specifying Telemetry Host-Initiated Log Page to the NVMe Controller with a Log Page Offset Lower value that is not a multiple of 512 bytes.

Observable Results:

1. Verify that NVMe Controller responds to the Get Log Page command with an error of Invalid Field in Command.

Case 14: **Telemetry Controller Initiated Valid Offset (M , OF-FYI)**

Test Procedure:

1. Check the Bit 3 of the LPA field Identify Controller Data Structure to determine of the DUT supports Telemetry Host-Initiated and Telemetry Controller–Initiated log pages. If Bit 3 is set to 0 this test is not applicable.
2. Configure the NVMe Host to issue a Get Log Page command specifying Telemetry Controller-Initiated Log Page to the NVMe Controller with a Log Page Offset Lower value that is a multiple of 512 bytes.

Observable Results:

1. Verify that NVMe Controller sends the requested Log Page following the Telemetry Controller-Initiated Log Page format defined in the NVMe specification.

Case 15: **Telemetry Controller Initiated Invalid Offset (M, OF-FYI)**

Test Procedure:

1. Check the Bit 3 of the LPA field Identify Controller Data Structure to determine of the DUT supports Telemetry Host-Initiated and Telemetry Controller–Initiated log pages. If Bit 3 is set to 0 this test is not applicable.
2. Configure the NVMe Host to issue a Get Log Page command specifying Telemetry Controller-Initiated Log Page to the NVMe Controller with a Log Page Offset Lower value that is not a multiple of 512 bytes.

Observable Results:

1. Verify that NVMe Controller responds to the Get Log Page command with an error of Invalid Field in Command.

Case 16: **Telemetry Host Initiated Valid 0 Offset Create=1 (M, OF-FYI)**

Test Procedure:

1. Check the Bit 3 of the LPA field Identify Controller Data Structure to determine of the DUT supports Telemetry Host-Initiated and Telemetry Controller–Initiated log pages. If Bit 3 is set to 0 this test is not applicable.
2. Configure the NVMe Host to issue a Get Log Page command specifying Telemetry Host-Initiated Log Page to the NVMe Controller with a Log Page Offset of 0, and the Create Telemetry Host-Initiated Data bit is set to 1.

Observable Results:

1. Verify that NVMe Controller sends the requested Log Page using the correct formatting of the first 512 bytes of the Telemetry Host-Initiated Log Page including the reserved bytes.

Case 17: **Data Units Written Does not Increment for Write Uncorrectable (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Identify Namespace Data Structure Command CNS=00 and record the value for LBA Data Size (LBADS) for LBA Format 0.
2. Configure the NVMe Host to issue a Identify Controller Data Structure Command CNS=01 and record the value for ONCS Bit 1. If ONCS Bit 1 is set to 0, then the DUT does not support the Write Uncorrectable Command and this test is not applicable. If ONCS Bit 1 is set 1, then proceed to the next step.
3. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value.
4. Perform 1000 NVMe Write Uncorrectable commands of LBADS bytes..
5. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value, which is expected to be unchanged.
6. Perform 1000 NVMe Write Uncorrectable commands of 2x LBADS bytes..
7. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value, which is expected to increase by 2x LBADS/512. (i.e. if LBADS is 512, then the Data Units Written value, which is expected to be unchanged.
8. Perform 1000 NVMe Write Uncorrectable commands of 4x LBADS bytes..
9. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value, which is expected to increase by 4x LBADS/512. (i.e. if LBADS is 512, then the Data Units Written value, which is expected to be unchanged.
10. Perform a Write operation to all LBAs that a Write Uncorrectable operation was performed to to enable those blocks to be read in future test operations.

Observable Results:

1. Verify that NVMe Controller reported the Data Units Written value that did not increase when the Write Uncorrectable commands were performed, and this was properly reflected in each SMART/Health Information Log Page returned by the DUT.

Case 18: **Persistent Event Log (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 4 of the Log Page Attributes field in the Identify Controller Data Structure. If Bit 4 is set to 0, then this test is not applicable.
2. Configure the NVMe Host to issue a Get Log Page Command for Log Identifier 0Dh “Persistent Event Log”.
3. Read the supported events bitmap returned in Log Page 0Dh, then cause the testing station to perform one of the events supported by the DUT. Common events supported may be :
    a. Set Feature
    b. Sanitize

   c. Format NVM
   d. Change Namespace
   e. Power on or Reset.
4. If no events that can be performed by the test station are supported, then this test is not applicable.
5. Configure the NVMe Host to issue a Get Log Page Command for Log Identifier 0Dh "Persistent Event Log".
6. Perform a Controller Level Reset.
7. Configure the NVMe Host to issue a Get Log Page Command for Log Identifier 0Dh "Persistent Event Log".

Observable Results:

1. Verify that all Get Log Page commands completed with status success.
2. Verify that the Log page contents indicated that a supported event was performed, after the testing station performed one of the supported events.
3. Verify that the Log Page contents were not reset after the Controller Level Reset.

Case 19: **Data Units Read Count – Verify (FYI, OF-FYI)**

Test Procedure:

1. Check ONCS bit 7 to determine if the DUT supports the Verify Command. If the Verify Command is not supported, this test is not applicable.
2. Configure the NVMe Host to issue a Identify Namespace Data Structure Command CNS=00 and record the value for LBA Data Size (LBADS) for LBA Format 0.
3. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value.
4. Perform 1000 Verify commands of LBADS bytes, using the same protection information as the namespace was formatted with.
5. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 1).
6. Perform 1000 Verify commands of 2x LBADS bytes, using the same protection information as the namespace was formatted with.
7. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 2).
8. Perform 1000 Verify commands of 4x LBADS bytes, using the same protection information as the namespace was formatted with.
9. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 4).

Observable Results:

1. Verify that NVMe Controller reported the Data Units Read value that increased by 7x LBADS/512 in accord with the NVMe Verify commands that were performed.

Case 20: **Data Units Read Count – Read Only (FYI, OF-FYI)**

Test Procedure:

1. Configure the NVMe Host to issue a Identify Namespace Data Structure Command CNS=00 and record the value for LBA Data Size (LBADS) for LBA Format 0.
2. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value.
3. Perform 1000 NVMe READ command of LBADS bytes.
4. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 1). .
5. Perform 1000 NVMe READ command of 2x LBADS bytes.

6. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 2).
7. Perform 1000 NVMe READ command of 4x LBADS bytes.
8. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Read value, which is expected to increase by LBADS/512. (i.e. if LBADS is 512, then the Data Units Read value should increase by 4).

Observable Results:

1. Verify that NVMe Controller reported the Data Units Read value that increased by 7x (LBADS/512) in accord with the NVMe READ commands that were performed.

Case 21: **Data Units Written Does not Increment for Write Zeroes (FYI, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Identify Namespace Data Structure Command CNS=00 and record the value for LBA Data Size (LBADS) for LBA Format 0.
2. Configure the NVMe Host to issue a Identify Controller Data Structure Command CNS=01 and record the value for ONCS Bit 1. If ONCS Bit 1 is set to 0, then the DUT does not support the Write Uncorrectable Command and this test is not applicable. If ONCS Bit 1 is set 1, then proceed to the next step.
3. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value.
4. Perform 1000 NVMe Write Zeroes commands of LBADS bytes.
5. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value, which is expected to be unchanged.
6. Perform 1000 NVMe Write Zeroes commands of 2x LBADS bytes.
7. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value , which is expected to be unchanged.
8. Perform 1000 NVMe Write Zeroes commands of 4x LBADS bytes.
9. Configure the NVMe Host to issue a Get Log Page command to LID 02h, SMART/Health Information (02h). and record the Data Units Written value , which is expected to be unchanged.

Observable Results:

1. Verify that NVMe Controller reported the Data Units Written value that did not increase when the Write Zeroes commands were performed, and this was properly reflected in each SMART/Health Information Log Page returned by the DUT.

Case 22: **Invalid LPOL offset (FYI, OF-FYI)**

Test Procedure:

1. Configure the NVMe Host to issue a Get Log Page command specifying each of the Mandatory Log Page Identifiers described in Table 3 and 4 above, to the NVMe Controller. For each mandatory Log Page requested, the LPOL field in the Get Log Page Request should be set to a value greater than the size of the log page requested.

Observable Results:

1. Verify that the Get Log Page command is aborted with status 'Invalid Field in Command'.

Case 23: **Invalid LPOU offset (M, OF-FYI)**

Test Procedure:

1. Configure the NVMe Host to issue a Get Log Page command specifying each of the Mandatory Log Page Identifiers described in Table 3 and 4 above, to the NVMe Controller. For each mandatory Log Page requested, the LPOU field in the Get Log Page Request should be set to a value greater than the size of the log page requested.

Observable Results:

1. Verify that the Get Log Page command is aborted with status 'Invalid Field in Command'.

Case 24: **Domain Identifier (FYI, OF-FYI)**

Test Procedure:

1. Configure the NVMe Host to query the Identifier Controller Data Structure MDS bit, Bit 10 (MDS) and Bit 4 (Endurance Groups) of the CTRATT field.
2. If the CTRATT Bit 4 (Endurance Groups) is set to 0, this test is not applicable.
3. Configure the NVMe Host to issue a Get Log Page command for the Endurance Group Log (LID=09h).

Observable Results:

1. Verify that if the MDS bit is set to 0, that the Domain Identifier in the Endurance Group Log page is set to 0 also.
2. Verify that if the MDS bit is set to 1, that the Domain Identifier in the Endurance Group Log page is set to a non zero value.

Possible Problems: None.

Test 1.4 – Create/Delete I/O Submission and Completion Queue Commands (M)

Purpose: To verify that an NVMe Controller can properly create and delete I/O Submission and Completion Queues.

References:

[1] NVMe Specification 5.3, 5.4, 5.5, 5.6, NVMe v1.3 ECN 001

Resource Requirements:

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

Last Modification: May 4, 2020

Discussion: The Create I/O Completion Queue command is used to create I/O Completion Queues while the Delete I/O Completion Queue command is used to delete I/O Submission Queues. Likewise, the Create I/O Submission Queue command is used to create I/O Submission Queues while the Delete I/O Submission Queue command is used to delete I/O Submission Queues. Host software shall ensure that any associated I/O Submission Queue is deleted prior to deleting a Completion Queue.

The Create I/O Submission Queue and Create I/O Completion Queue commands use the PRP Entry 1, Command Dword 10, and Command Dword 11 fields. The Delete I/O Submission Queue and Delete I/O Completion Queue commands use the Command Dword 10. All other command specific fields are reserved.

A completion queue entry is posted to the Admin Completion Queue when the specified queue has been created or deleted.

Test Setup: See Appendix A.

Case 1: **Basic Operation (M)**

Test Procedure:

1. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller.
2. Configure the NVMe Host to issue a Create I/O Submission Queue command to the controller.
3. Configure the NVMe Host to issue 10 Read commands assigned to the queues created in steps 1 and 2 to the controller.
4. Configure the NVMe Host to issue a Delete I/O Submission Queue command specifying the Submission Queue ID of the queue created in step 2 to the controller.
5. Configure the NVMe Host to issue a Delete I/O Completion Queue command specifying the Completion Queue ID of the queue created in step 1 to the controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the queues are properly created and deleted by reading the completion queue entries posted for each command issued by the NVMe Host (all statuses should be Success).
3. Verify that all received responses have all Reserved fields set to 0.

Case 2: **Create I/O Completion Queue with QID=0h, exceeds Number of Queues reported, Identifier Already in Use (M)**

Test Procedure:

1. Configure the NVMe Host to issue a Get Features command for the Feature Identifier 07h, Number of Queues, record the value reported.

2. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller, with QID=0h.
3. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller with QID=greater than the value reported in the Number of Queues field in the Get Feature response. If the DUT supports the maximum possible number of Queues than this test case is not applicable.
4. Configure the NVMe Host to issue 2 Create I/O Completion Queue commands with identical QID. The first command is expected to complete successfully, the second is expected to return an error of Invalid Queue Identifier.

Observable Results:

1. Verify that after the completion of each command, the controller returns an error of Invalid Queue Identifier

Case 3: **Delete I/O Completion Queue before deleting Corresponding Submission Queue (M)**

Test Procedure:

1. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller.
2. Configure the NVMe Host to issue a Create I/O Submission Queue command to the controller.
3. Configure the NVMe Host to issue 10 Read commands assigned to the queues created in steps 1 and 2 to the controller.
4. Configure the NVMe Host to issue a Delete I/O Completion Queue command specifying the Completion Queue ID of the queue created in step 1 to the controller.
5. Configure the NVMe Host to issue a Delete I/O Submission Queue command specifying the Submission Queue ID of the queue created in step 2 to the controller.
6. Configure the NVMe Host to issue a Delete I/O Completion Queue command specifying the Completion Queue ID of the queue created in step 1 to the controller.

Observable Results:

1. Verify that the Delete I/O Completion Queue command in step 4, caused the controller to returns an error of Invalid Queue Deletion.

Case 4: **Create I/O Completion Queue with Invalid Queue Size (M)**

Test Procedure:

1. Configure the NVMe Host to read the Capabilities Register CAP.MQES field to obtain the Maximum Queue Size supported by the DUT. Record the value.
2. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller with a QSIZE of 0h.
3. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller with a QSIZE of greater than the Queue size (MQES) supported by the DUT.

Observable Results:

1. Verify that in each case the Create I/O Completion Queue, caused the controller to returns an error of Invalid Queue Size.

Case 5: **Create I/O Submission Queue with Invalid Queue Size (M)**

Test Procedure:

1. Configure the NVMe Host to read the Capabilities Register CAP.MQES field to obtain the Maximum Queue Size supported by the DUT. Record the value.
2. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller.
3. Configure the NVMe Host to issue a Create I/O Submission Queue command to the controller with a QSIZE of 0h.
4. Configure the NVMe Host to issue a Create I/O Submission Queue command to the controller with a QSIZE of greater than the Queue size (MQES) supported by the DUT.

Observable Results:

1. Verify that in each case the Create I/O Submission Queue, caused the controller to returns an error of Invalid Queue Size.

Case 6: **Create I/O Submission Queue Physically Contiguous (M)**

Test Procedure:

1. Configure the NVMe Host to read the Capabilities Register CAP.CQR field. If CAP.CQR is set to zero, this test case is not applicable. If CAP.CQR is set to 1, proceed to the next step.
2. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller.
3. Configure the NVMe Host to issue a Create I/O Submission Queue command to the controller with the PC bit set to 0.

Observable Results:

1. Verify that the transmitted Create I/O Submission Queue, caused the controller to returns an error of Invalid Field in Command.

Case 7: **Create I/O Submission Queue Invalid CQID of 0h (M)**

Test Procedure:

1. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller.
2. Configure the NVMe Host to issue a Create I/O Submission Queue command to the controller with the CQID set to 0h.

Observable Results:

1. Verify that the transmitted Create I/O Submission Queue with CQID of 0h, caused the controller to returns an error of 1h - Invalid Queue Identifier.

Case 8: **Create I/O Completion Queue Invalid Interrupt Vector (M)**

Test Procedure:

1. Configure the NVMe Host to read the MSICAP.MC.MME field and the MSIXCAP.MXC.TS.
2. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller with an Interrupt Vector (IV) value greater than the number of messages supported by the DUT as indicated in either MSICAP.MC.MME or MSIXCAP.MXC.TS.

Observable Results:

1. Verify that the transmitted Create I/O Completion Queue, caused the controller to return an error of Invalid Interrupt Vector.

Case 9: **Create I/O Submission Queue Invalid CQID Outside Supported Range (FYI)**

Test Procedure:

1. Perform a Get Feature command for Feature ID 07h to obtain the NCQA value.
2. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller.
3. Configure the NVMe Host to issue a Create I/O Submission Queue command to the controller with the CQID set to value outside the range supported by the DUT, as indicated in NCQA.

Observable Results:

1. Verify that the transmitted Create I/O Submission Queue with CQID outside the supported range, caused the controller to return an error of 1h - Invalid Queue Identifier.

Case 10: **Create I/O Submission Queue Invalid CQID within supported range, but Queue not created (M)**

Test Procedure:

1. Perform a Get Feature command for Feature ID 07h to obtain the NCQA value.
2. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller.
3. Configure the NVMe Host to issue a Create I/O Submission Queue command to the controller with the CQID set to a value which is within the supported range (as indicated by NCQA) but for a queue which has not been created.

Observable Results:

1. Verify that the transmitted Create I/O Submission Queue with CQID with an invalid value within the supported range as indicated in NCQA (3h is used in some test implementations), caused the controller to return an error of 0h - Completion Queue Invalid.

Case 11: **Set Feature After Queues Created (FYI)**

Test Procedure:

1. Configure the NVMe Host to issue a Create I/O Completion Queue command to the controller.
2. Configure the NVMe Host to issue a Create I/O Submission Queue command to the controller.
3. Configure the NVMe Host to issue 10 Read commands assigned to the queues created in steps 1 and 2 to the controller.
4. Configure the NVMe Host to issue a Delete I/O Submission Queue command specifying the Submission Queue ID of the queue created in step 2 to the controller.
5. Configure the NVMe Host to issue a Delete I/O Completion Queue command specifying the Completion Queue ID of the queue created in step 1 to the controller.
6. Configure the NVMe Host to issue a Set Feature Command for the Number of Queues Feature, FID=07h.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the queues are properly created and deleted by reading the completion queue entries posted for each command issued by the NVMe Host (all statuses should be Success).
3. Verify that all received responses have all Reserved fields set to 0.
4. Verify that the Set Feature Command failed with status ‘Command Sequence Error’.

Possible Problems: None.

Test 1.5 – Abort Command (M)

Purpose: To determine the conditions under which an NVMe system can successfully abort a given command.

References:

[1] NVMe Specification 5.1, NVMe v1.3 ECN 003

Resource Requirements:

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

Last Modification: December 1, 2019

Discussion: The Abort command is used to abort a specific command previously issued to the Admin Submission Queue or an I/O Submission Queue. Host software may have multiple Abort commands outstanding, subject to the constraints of the Abort Command Limit indicated in the Identify Controller data structure. An abort is a best effort command; the command to abort may have already completed, currently be in execution, or may be deeply queued. It is implementation specific if/when a controller chooses to complete the command when the command to abort is not found. Whether the command was successfully aborted or not is determined by examining bit 0 of command Dword 0 of the completion queue entry for the Abort command. If the command was successfully aborted, then bit 0 of Dword 0 is cleared to '0'. If the command was not aborted, then bit 0 of Dword 0 is set to '1'.The completion queue entry for the Abort command is only posted to the Admin Completion Queue after the completion queue entry for the command specified in the Abort command has been posted to its corresponding Admin or I/O Completion Queue.

The Abort command uses the Command Dword 10 field. All other command specific fields are reserved.

Since the Abort Command is a best effort command, this test is designed to determine under what conditions a command can successfully be aborted, and is therefore considered an informative test.

Test Setup: See Appendix A.

Case 1: **Abort I/O Command (M)**

Test Procedure:

1. Configure the NVMe Host to issue 10 Read commands to the controller.
2. Configure the NVMe Host to issue an Abort command to the controller specifying the CID of one of the issued Read commands.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Determine whether the requested command was successfully aborted by examining bit 0 of Dword 0 of the completion queue entry for the Abort command.
    a. If bit 0 of Dword 0 is cleared to '0' in the completion queue entry for the Abort command, the command was aborted. If the READ command was successfully aborted, verify that the status of the aborted command is Command Abort Requested (07h), and that the Completion Queue Entry for the aborted READ command is posted to the I/O Completion Queue *before* the Completion Queue Entry of the Abort command is posted to the Admin Completion Queue.
    b. If bit 0 of Dword 0 is set to 1, the command was not aborted. Verify that a completion Queue Entry for the Abort command is posted to the Admin Completion Queue.
3. Verify that all received responses have all Reserved fields set to 0.

Case 2: **Abort Admin Command (FYI)**

Test Procedure:

1. Configure the NVMe Host to issue 10 Get Log Page commands to the controller. Any supported Log Page ID can be used.
2. Configure the NVMe Host to issue an Abort command to the controller specifying the CID of one of the issued Get Log Page commands.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Determine whether the requested command was successfully aborted by examining bit 0 of Dword 0 of the completion queue entry for the Abort command.
    a. If bit 0 of Dword 0 is cleared to '0' in the completion queue entry for the Abort command, the command was aborted. If the Get Log Page command was successfully aborted, verify that the status of the aborted command is Command Abort Requested (07h), and that the Completion Queue Entry for the aborted Get Log Page command is posted to the Admin Completion Queue before the Completion Queue Entry of the Abort command is posted to the Admin Completion Queue.
    b. If bit 0 of Dword 0 is set to 1, the command was not aborted. Verify that a completion Queue Entry for the Abort command is posted to the Admin Completion Queue.
3. Verify that all received responses have all Reserved fields set to 0.

Possible Problems: Cannot test ABORT Limit Exceeded execution time limit (there's a chance the 1st ABORT finishes before the last one)

Test 1.6 – Format NVM Command (M, OF)

Purpose: To verify that an NVMe Controller can properly execute the Format NVM command.

References:

[1] NVMe Specification 5.23, NVMe v1.3 ECN 002, NVMe v1.3 ECN 005

Resource Requirements:

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

Last Modification: April 8, 2019

Discussion: The Format NVM command is used to low level format the NVM media. This is used when the host wants to change the LBA data size and/or metadata size. A low level format may destroy all data and metadata associated with all namespaces or only the specific namespace associated with the command. After the Format NVM command successfully completes, the controller shall not return any user data that was previously contained in an affected namespace.

The settings specified in the Format NVM command are reported as part of the Identify Namespace data structure. If the controller supports multiple namespaces, then the host may specify the value of FFFFFFFFh for the namespace in order to apply the format operation to all namespaces accessible by the controller regardless of the value of the Format NVM Attribute field in the Identify Controller data structure.

The Format NVM command uses the Command Dword 10 field. All other command specific fields are reserved.

Test Setup: See Appendix A.

Case 1: **Valid LBAF, SES=000b (M, OF)**

Test Procedure:

1. Check the OACS field of the Identify Controller Data structure to determine if the DUT supports the Format NVM command. If the command is not supported this test is not applicable.
2. Record the values in the DPS field of the Identify Namespace Data structure.
3. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to an LBA range.
4. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step3.
5. Configure the NVMe Host to issue a Format NVM Command with an LBAF Supported by the DUT, SES=000b and FFFFFFFFh for the namespace ID.
6. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step3.
7. Perform a Format NVM operation to restore the DPS settings for the Namespace that were recorded in Step 2.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the same data pattern which was written to the NVM in step 3 is read from the NVM in step 4.
3. Verify that the data pattern written in step 4 is not returned by the controller in step 6.
4. Verify that all received responses have all Reserved fields set to 0.

Case 2: **Valid LBAF, SES=001b (M, OF-FYI)**

Test Procedure:

1. Check the OACS field of the Identify Controller Data structure to determine if the DUT supports the Format NVM command. If the command is not supported this test is not applicable.

2. Record the values in the DPS field of the Identify Namespace Data structure
3. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to an LBA range.
4. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step3.
5. Configure the NVMe Host to issue a Format NVM Command with an LBAF Supported by the DUT, SES=001b and FFFFFFFFh for the namespace ID.
6. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step3.
7. Perform a Format NVM operation to restore the DPS settings for the Namespace that were recorded in Step 2.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the same data pattern which was written to the NVM in step 3 is read from the NVM in step 4.
3. Verify that the data pattern written in step 3 is not returned by the controller in step 6.
4. Verify that all received responses have all Reserved fields set to 0.

Case 3: **Valid LBAF, SES=010b (M, OF-FYI)**

Test Procedure:

1. Check the OACS field of the Identify Controller Data structure to determine if the DUT supports the Format NVM command. If the command is not supported this test is not applicable.
2. Record the values in the DPS field of the Identify Namespace Data structure
3. Check Bit 2 of the FNA field of the Identify Controller Data structure to determine if the DUT supports cryptographic erase. If the DUT does not support cryptographic erase this test case is not applicable.
4. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to an LBA range.
5. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step4.
6. Configure the NVMe Host to issue a Format NVM Command with an LBAF Supported by the DUT, SES=010b and FFFFFFFFh for the namespace ID.
7. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step4.
8. Perform a Format NVM operation to restore the DPS settings for the Namespace that were recorded in Step 2.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the same data pattern which was written to the NVM in step 4 is read from the NVM in step 5.
3. Verify that the data pattern written in step 4 is not returned by the controller in step 7.
4. Verify that all received responses have all Reserved fields set to 0.

Case 4: **Valid LBAF, SES=111b (reserved value) (M, OF-FYI)**

Test Procedure:

1. Check the OACS field of the Identify Controller Data structure to determine if the DUT supports the Format NVM command. If the command is not supported this test is not applicable.
2. Record the values in the DPS field of the Identify Namespace Data structure
3. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to an LBA range.
4. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step 4.

5. Configure the NVMe Host to issue a Format NVM Command with an LBAF Supported by the DUT, SES=111b and FFFFFFFFh for the namespace ID.
6. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step 4.
7. Perform a Format NVM operation to restore the DPS settings for the Namespace that were recorded in Step 2.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the same data pattern which was written to the NVM in step 3 is read from the NVM in step 4.
3. Verify that the data pattern written in step 3 *is* returned by the controller in step 6.
4. Verify that the Format NVM command completes with error status code Invalid Field (02h).
5. Verify that all received responses have all Reserved fields set to 0.

Case 5: **Unsupported LBAF, SES=000b (M, OF-FYI)**

Test Procedure:

1. Check the OACS field of the Identify Controller Data structure to determine if the DUT supports the Format NVM command. If the command is not supported this test is not applicable.
2. Check the NLBAF field in the Identify Namespace Data Structure to determine the number of LBA formats supported by the DUT. If NLBAF is set to 16, then LBAF cannot be set to an invalid value and this test is not applicable.
3. Record the values in the DPS field of the Identify Namespace Data structure.
4. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to an LBA range.
5. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step 4.
6. Configure the NVMe Host to issue a Format NVM Command with an invalid LBAF.
7. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step 4.
8. Perform a Format NVM operation to restore the DPS settings for the Namespace that were recorded in Step 2.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the same data pattern which was written to the NVM in step 4 is read from the NVM in step 5.
3. Verify that the data pattern written in step 4 *is* returned by the controller in step 7.
4. Verify that the Format NVM command completes with error status code Invalid Format (0Ah).
5. Verify that all received responses have all Reserved fields set to 0.

Case 6: **Unsupported LBAF, SES=111b (reserved value) (M, OF-FYI)**

Test Procedure:

1. Check the OACS field of the Identify Controller Data structure to determine if the DUT supports the Format NVM command. If the command is not supported this test is not applicable.
2. Check the NLBAF field in the Identify Namespace Data Structure to determine the number of LBA formats supported by the DUT. If NLBAF is set to 16, then LBAF cannot be set to an invalid value and this test is not applicable.
3. Record the values in the DPS field of the Identify Namespace Data structure
4. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to an LBA range.
5. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step 4.
6. Configure the NVMe Host to issue a Format NVM Command with an invalid LBAF and SES=111b.

7. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step 4.
8. Perform a Format NVM operation to restore the DPS settings for the Namespace that were recorded in Step 2.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the same data pattern which was written to the NVM in step 4 is read from the NVM in step 5.
3. Verify that the data pattern written in step 4 *is* returned by the controller in step 7.
4. Verify that the Format NVM command completes with error status code of Invalid Field (02h) or Invalid Format (0Ah).
5. Verify that all received responses have all Reserved fields set to 0.

Case 7: **Valid LBAF, SES=000b, PI is non-zero (M)**

Test Procedure:

1. Check the OACS field of the Identify Controller Data structure to determine if the DUT supports the Format NVM command. If the command is not supported this test is not applicable.
2. Check the DPC field of the Identify Namespace Data Structure to determine if the DUT supports End to End Data Protection and this test is not applicable.
3. Record the values in the DPS field of the Identify Namespace Data structure
4. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to an LBA range.
5. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in the previous step.
6. Configure the NVMe Host to issue a Format NVM Command with LBAF with non-zero metadata size supported by the DUT, SES=000b and FFFFFFFFh for the namespace ID, and PI=001b.
7. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step 4.
8. Perform a Format NVM operation to restore the DPS settings for the Namespace that were recorded in Step 2.
9. Repeat for PI=010b and PI=011b.

Observable Results:

1. If End to End Data Protection is supported:
    a. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
    b. Verify that the same data pattern which was written to the NVM in step 4 is read from the NVM in step 5.
    c. Verify that the data pattern written in step 4 is not returned by the controller in step 7.
2. If End to End Data Protection is not supported, verify that the command completed with error status code Invalid Field (02h).
3. Verify that all received responses have all Reserved fields set to 0.

Case 8: **Valid LBAF, SES=000b, PI=Type 3 (FYI, OF-FYI)**

Test Procedure:

1. Check the OACS field of the Identify Controller Data structure to determine if the DUT supports the Format NVM command. If the command is not supported this test is not applicable.
2. Record the values in the DPS field of the Identify Namespace Data structure
3. Check the DPC field to determine the Protection Information types supported by the DUT. If the DUT does not support end to end data protection Type 3, then this case is not applicable.
4. Configure the NVMe Host to issue a Format NVM Command with an LBAF with non-zero metadata size supported by the DUT, SES=000b, PIL=1 or 0 depending on value supported by the DUT, PI=Type 3 and a valid namespace ID.

5. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was formatted in step 4, and bit 0 of PRCHK field set to 1.
6. Configure the NVMe Host to issue a Compare command to the controller specifying the same LBA range which was formatted in step 4, and bit 0 of PRCHK field set to 1.
7. Configure the NVMe Host to issue a Write command to the controller specifying the same LBA range which was formatted in step 4, and bit 0 of PRCHK field set to 1.
8. Configure the NVMe Host to issue a Write Zeroes command to the controller specifying the same LBA range which was formatted in step 4, and bit 0 of PRCHK field set to 1.
9. Perform a Format NVM operation to restore the DPS settings for the Namespace that were recorded in Step 2.

Observable Results:

1. Verify that the Read, Compare, Write, and Write Zeroes commands all completed with status Invalid Protection Information. This functionality may be implemented by DUTs that support NVMe v1.3 or earlier, but must be implemented by DUTs that support NVMe v1.4 or later.
2. Verify that all received responses have all Reserved fields set to 0.

Case 9: **NSID=FFFFFFFFh, no attached namespaces, SES=000 (FYI, OF-FYI)**

Test Procedure:

1. Check the OACS field and FNA field of the Identify Controller Data structure. If OACS field indicates that the DUT does not support the Format NVM command this test is not applicable. If bit 0 of the FNA field is set to 0, this test is not applicable.
2. Record the values in the DPS field of the Identify Namespace Data structure.
3. Configure the NVMe Host to issue a Write command to an attached namespace in order to write a known data pattern to an LBA range. If there is no namespaces attached, create a namespace and attach it.
4. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step 4, then detach all namespaces.
5. Configure the NVMe Host to issue a Format NVM Command with an LBAF Supported by the DUT, SES=000b, NSID=FFFFFFFFh. Reattach the namespace written to in step 3.
6. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step 4.
7. Perform a Format NVM operation to restore the DPS settings for the Namespace that were recorded in Step 2.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the same data pattern which was written to the NVM in step 3 is read from the NVM in step 4.
3. Verify that the data pattern written in step 3 is not returned by the controller in step 6.
4. Verify that all received responses have all Reserved fields set to 0.

Case 10: **NSID=FFFFFFFFh, no attached namespaces, SES≠000 (FYI, OF-FYI)**

Test Procedure:

1. Check the OACS field and FNA field of the Identify Controller Data structure. If OACS field indicates that the DUT does not support the Format NVM command this test is not applicable. If bit 1 of the FNA field is set to 0, this test is not applicable.
2. Record the values in the DPS field of the Identify Namespace Data structure.
3. Configure the NVMe Host to issue a Write command to an attached namespace in order to write a known data pattern to an LBA range.
4. Configure the NVMe Host to issue a Read command to the same namespace specifying the same LBA range which was written in step 4, then detach the namespace. Detach all namespaces.

5. Configure the NVMe Host to issue a Format NVM Command with an LBAF Supported by the DUT, SES≠000b (check for all supported non zero values for SES), NSID=FFFFFFFFh. Reattach the namespace.
6. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA range which was written in step4.
7. Perform a Format NVM operation to restore the DPS settings for the Namespace that were recorded in Step 2.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the same data pattern which was written to the NVM in step 3 is read from the NVM in step 4.
2. Verify that the data pattern written in step 3 is not returned by the controller in step 6.
3. Verify that all received responses have all Reserved fields set to 0.

Case 11: **NSID=FFFFFFFFh Supported (FYI, OF-FYI)**

Test Procedure:

1. Check the OACS field and FNA field of the Identify Controller Data structure. If OACS field indicates that the DUT does not support the Format NVM command this test is not applicable. If bit 3 of the FNA field is set to 1, this test is not applicable.
2. Configure the NVMe Host to issue a Format NVM Command with an LBAF Supported by the DUT, NSID=FFFFFFFFh.

Observable Results:

1. Verify that the Format Command completes with status Success.

Case 12: **NSID=FFFFFFFFh Not Supported (FYI, OF-FYI)**

Test Procedure:

1. Check the OACS field and FNA field of the Identify Controller Data structure. If OACS field indicates that the DUT does not support the Format NVM command this test is not applicable. If bit 3 of the FNA field is set to 0, this test is not applicable.
2. Configure the NVMe Host to issue a Format NVM Command with an LBAF Supported by the DUT, NSID=FFFFFFFFh.

Observable Results:

1. Verify that the Format Command completes with status Invalid Field in Command.

Possible Problems: None known.

Test 1.7 – Asynchronous Events (M, OF)

Purpose: To verify that an NVMe Controller can properly report asynchronous events to the host.

References:

[1] NVMe Specification 5.2

Resource Requirements:

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

Last Modification: November 20, 2018

Discussion: Asynchronous events are used to notify host software of status, error, and health information as these events occur. To enable asynchronous events to be reported by the controller, host software needs to submit one or more Asynchronous Event Request commands to the controller. The controller specifies an event to the host by completing an Asynchronous Event Request command. Host software should expect that the controller may not execute the command immediately; the command should be completed when there is an event to be reported.

Test Setup: See Appendix A.

Case 1: **Asynchronous Event Request Command (M)**

The Asynchronous Event Request command is submitted by host software to enable the reporting of asynchronous events from the controller. This command has no timeout. The controller posts a completion queue entry for this command when there is an asynchronous event to be reported to the host. If Asynchronous Event Request commands are outstanding when the controller is reset, the commands are aborted.

Host software may submit multiple Asynchronous Event Request commands to reduce event reporting latency. The total number of simultaneously outstanding Asynchronous Event Request commands is limited by the Asynchronous Event Request Limit specified in the Identify Controller data structure.

If the controller needs to report an event and there are no outstanding Asynchronous Event Request commands, the controller should send a single notification of that Asynchronous Event Type when an Asynchronous Event Request command is received. If a Get Log Page command clears the event prior to receiving the Asynchronous Event Request command or if a power off condition occurs, then a notification is not sent.

The following event types are defined in the NVMe Specification:

- Error Event
- SMART / Health Event
- Notice Events
- I/O Command Set Specification (NVM Command Set) Events:
    - Reservation Log Page Available Event
- Vendor Specific Events

All command specific fields are reserved.

Test Procedure:

1. For each event type described above:
    a. Configure the NVMe Host to issue an Asynchronous Event Request command to the NVMe Controller.
    b. Configure the NVMe Host to cause conditions for generating the event for the NVMe Controller as described in the NVMe Specification.

Observable Results:

1. Verify that the NVMe Controller does not post a completion queue entry to the Admin Completion Queue for the Asynchronous Event Request command until after the event is generated.
2. Verify that the completion queue entry for the Asynchronous Event Request command is properly formatted and contains information appropriate for the event in Dword 0 as described in the NVMe Specification.
3. Verify that all received responses have all Reserved fields set to 0.
4.

Case 2: **Outstanding Commands Aborted after Reset (M, OF)**

If Asynchronous Event Request commands are outstanding when the controller is reset, the commands are aborted.

Test Procedure:

1. Configure the NVMe Host to issue an Asynchronous Event Request command to the NVMe Controller.
2. Configure the NVMe Host to initiate a Controller Level Reset for the NVMe Controller.

Observable Results:

1. Verify that after the Controller Level Reset is initiated, if a completion queue entry is posted by the NVMe Controller to the Admin Completion Queue and the status for the command is 07h Command Abort Requested. If no completion queue entry is posted, then this test is not applicable.
2. Verify that all received responses have all Reserved fields set to 0.

Case 3: **Clearing Events (IP)**

If the controller needs to report an event and there are no outstanding Asynchronous Event Request commands, the controller should send a single notification of that Asynchronous Event Type when an Asynchronous Event Request command is received. If a Get Log Page command clears the event prior to receiving the Asynchronous Event Request command or if a power off condition occurs, then a notification is not sent.

When the controller posts a completion queue entry for an outstanding Asynchronous Event Request command and thus reports an asynchronous event, subsequent events of that event type are automatically masked by the controller until the host clears that event. An event is cleared by reading the log page associated with that event using the Get Log Page command.

Test Procedure:

1. For each event type associated with a log page:
    a. Configure the NVMe Host to cause conditions for generating the event for the NVMe Controller as described in the NVMe Specification.
    b. Configure the NVMe Host to issue a Get Log Page command to the NVMe Controller for the log page associated with the event in order to clear that event.
    c. Configure the NVMe Host to issue an Asynchronous Event Request command to the NVMe Controller.
    d. After a timeout period of 2 seconds, configure the NVMe Host to clear the outstanding Asynchronous Event Request command by initiating a Controller Level Reset for the NVMe Controller.

Observable Results:

1. Verify that after the completion of each Get Log Page command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the NVMe Controller does not post a completion queue entry to the Admin Completion Queue for the Asynchronous Event Request command until after the Controller Level Reset and that the status indicates that the command was aborted and contains no information about the event in Dword 0.
3. Verify that all received responses have all Reserved fields set to 0.

Case 4: **Masking Events (M, OF)**

The Asynchronous Event Configuration feature controls the events that trigger an asynchronous event notification to the host. This Feature may be used to disable reporting events in the case of a persistent condition. The attributes are indicated in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in for Command Dword 11 of the Set Features command are returned in Dword 0 of the completion queue entry for that command.

Test Procedure:

1. For each event which may be disabled with the Asynchronous Event Configuration feature:
    a. Configure the NVMe Host to issue a Set Features command to the NVMe Controller with feature identifier 0Bh (Asynchronous Event Configuration) and formatted to disable the event.
    b. Configure the NVMe Host to issue an Asynchronous Event Request command to the NVMe Controller.
    c. Configure the NVMe Host to cause conditions for generating the event for the NVMe Controller as described in the NVMe Specification.
    d. After a timeout period of 2 seconds, configure the NVMe Host to clear the outstanding Asynchronous Event Request command by initiating a Controller Level Reset for the NVMe Controller.

Observable Results:

1. Verify that after the completion of each Set Features command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the NVMe Controller does not post a completion queue entry to the Admin Completion Queue for the Asynchronous Event Request command until after the Controller Level Reset and that the status indicates that the command was aborted and contains no information about the event in Dword 0.
3. Verify that all received responses have all Reserved fields set to 0.

Possible Problems: Some events are optional as reported by the Optional Asynchronous Events Supported (OAES) field of the Identify Controller data structure or by other means and are therefore only tested if those features are supported.

Test 1.8 – Get Feature Select (M)

Purpose: To verify that an NVMe Controller can properly execute a Get Features command with the Select Field set.

References:

[1] NVMe Specification 5.9.1

Resource Requirements:

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

Last Modification: June 9, 2016

Discussion: A Select field set to 000b (i.e., current) returns the current operating attribute value for the Feature Identifier specified.
A Select field set to 001b (i.e., default) returns the default attribute value for the Feature Identifier specified.

A Select field set to 010b (i.e., saved) returns the last saved attribute value for the Feature Identifier specified (i.e., the last Set Features command completed without error, with the Save bit set to '1' for the Feature Identifier specified.)

A Select field set to 011b (i.e., supported capabilities) returns the capabilities supported for this Feature Identifier. The capabilities supported are returned in Dword 0 of the completion entry of the Get Features command.

A completion queue entry is posted to the Admin Completion Queue when the controller has completed returning any attributes associated with the Feature. Depending on the Feature Identifier, Dword 0 of the completion queue entry may contain feature information

Test Setup: See Appendix A.

Test Procedure:

1. For each of the features described in Table 1 and Table 2 (see Test 1.2):
    a. Configure the NVMe Host to issue a Get Features command indicating a Select Field of 000b.
    b. Configure the NVMe Host to issue a Get Features command indicating a Select Field of 001b.
    c. Configure the NVMe Host to issue a Set Features command, to specify a new value for each FID. Next, configure the NVMe Host to issue a Get Features command indicating a Select Field of 010b.
    d. Configure the NVMe Host to issue a Get Features command indicating a Select Field of 011b.

Observable Results:

1. For Step 1a, verify that the DUT returns the current operating attribute value for the Feature Identifier specified.
2. For Step 1b, verify that the DUT returns the the default attribute value for the Feature Identifier specified.
3. For Step 1c, verify that the DUT returns the last saved attribute value for the Feature Identifier Specified. This must match the value assigned in Step 1c.
4. For Step 1d, verify that the DUT returns returns the capabilities supported for this Feature Identifier. The capabilities supported are returned in Dword 0 of the completion entry of the Get Features command.

Possible Problems: None.

Test 1.9 – Feature Saved Across Reset (M)

Purpose: To verify that an NVMe Controller can properly save a Feature value set by a Host across a reset.

References:

[1] NVMe Specification 7.8

Resource Requirements:

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

Last Modification: April 2, 2019

Discussion: If bit 4 is set to '1' in the Optional NVM Command Support field of the Identify Controller data structure, then for each Feature, there are three settings: default, saveable, and current.

The saveable value is the value that the Feature has after a power on or reset event. The controller may not support a saveable value for a Feature; this is discovered by using the 'supported capabilities' value in the Select field in Get Features. If the controller does not support a saveable value for a Feature, then the default value is used after a power on or reset event. The current value is the value actively in use by the controller for a Feature after a Set Features command completes.

This test is only applicable to devices which set the Save field (Bit 4) of the ONCS field to 1.

Test Setup: See Appendix A.

Test Procedure:

1. Check bit 4 of the ONCS field. If bit 4 is set to 1, proceed to the next step. If not the test cannot be performed.
2. For each of the features described in Table 1 and Table 2 (see Test 1.2):
    a. Determine which features the DUT indicates as both 'Saveable' and 'Changeable'.
    b. Configure the NVMe Host to issue a Get Features command indicating a Select Field of 000b, to obtain the current operating value of the Feature.
    c. Configure the NVMe Host to issue a Set Features command with the Save bit set to 1 (SV=1), to specify a new value for each FID that the DUT indicates as both 'Saveable' and 'Changeable'.
    d. Configure the NVMe Host to issue a Get Features command indicating a Select Field of 000b, to obtain the current operating value of the Feature.
    e. Perform an NVMe Subsystem Reset (NSSR) or another supported reset type.
    f. Configure the NVMe Host to issue a Get Features command indicating a Select Field of 000b, to obtain the current operating value of the Feature.

Observable Results:

1. For each Feature tested, verify that the value reported in Step 2e, matches the value set by the Host in Step 2b. prior to the reset.

Possible Problems: Not all devices will support NSSR, and other reset methods may be necessary.

Test 1.10 – Device Self-test Short Operation (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly perform a Device Self-test Short Operation.

References:

[1] NVMe Specification 5.8

Resource Requirements:

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

Last Modification: December 14, 2017

Discussion: The Device Self-test command is used to start a device self-test operation or abort a device self-test Operation. The device self-test operation is performed by the controller that the Device Self-test command was submitted to.

Test Setup: See Appendix A.

Case 1: **Namespace Test Action = 00000000h, STC=1h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0.
3. Configure the NVMe Host to send a Device Self-test command with Namespace Test Action set to 00000000h and STC set to 1h (Short device self-test operation).
4. Before the Device Self-test operation finishes, send a Get Log Page command to LID 06h (Device Self-test). Note that since the Device Self Test operation can be performed in the background, a command completion may be received before the Device Self Test operation finishes. Repeat this step until the returned log page indicates that the operation has finished.
5.
6. Report to the user the time elapsed between sending of the Device Self-test command and the operation finishing.

Observable Results:

1. Using the Log Page info returned in response to the first Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 1h when the operation was processing.
2. Verify that the Device Self-test command completes successfully.
3. Using the Log Page info returned in response to the second Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
4. Verify that all received responses have all Reserved fields set to 0.

Case 2: **Namespace Test Action = 00000001h-FFFFFFFEh, STC=1h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.

2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to all value between 00000001h and FFFFFFFEh that represent a valid Namespace on the DUT, with STC set to 1h (Short device self-test operation).
4. Before the Device Self-test operation finishes, send a Get Log Page command to LID 06h (Device Self-test). Note that since the Device Self Test operation can be performed in the background, a command completion may be received before the Device Self Test operation finishes. Repeat this step until the returned log page indicates that the operation has finished.
5.
6. Report to the user the time elapsed between sending of the Device Self-test command and the operation finishing.

Observable Results:

1. Using the Log Page info returned in response to the first Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 1h when the operation was processing.
2. Verify that the Device Self-test command completes successfully.
3. Using the Log Page info returned in response to the second Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
4. Verify that all received responses have all Reserved fields set to 0.

Case 3: **Namespace Test Action = FFFFFFFFh, STC=1h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to FFFFFFFFh, with STC set to 1h (Short device self-test operation).
4. Before the Device Self-test operation finishes, send a Get Log Page command to LID 06h (Device Self-test). Note that since the Device Self Test operation can be performed in the background, a command completion may be received before the Device Self Test operation finishes. Repeat this step until the returned log page indicates that the operation has finished.
5.
6. Report to the user the time elapsed between sending of the Device Self-test command and the operation finishing.

Observable Results:

1. Using the Log Page info returned in response to the first Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 1h when the operation was processing.
2. Verify that the Device Self-test command completes successfully.
3. Using the Log Page info returned in response to the second Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
4. Verify that all received responses have all Reserved fields set to 0.

Case 4: **Namespace Test Action = Invalid Namespace, STC=1h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to an Invalid Namespace, with STC set to 1h (Short device self-test operation).
4. Send a Get Log Page command to LID 06h (Device Self-test). Note that since the Device Self Test operation can be performed in the background, a command completion may be received before the Device Self Test operation finishes. Repeat this step until the returned log page indicates that the operation has finished.

Observable Results:

1. Verify that the Device Self-test command returns status Invalid Namespace or Format in Command.
2. Using the Log Page info returned in response to the second Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.

Case 5: **Namespace Test Action = Inactive Namespace, STC=1h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Configure the Testing Station acting as a host to Check Bit 3 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports Namespace Management. If Bit 3 is not set to 1, this test is not applicable.
3. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
4. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to an Inactive Namespace, with STC set to 1h (Short device self-test operation).

Send a Get Log Page command to LID 06h (Device Self-test). Note that since the Device Self Test operation can be performed in the background, a command completion may be received before the Device Self Test operation finishes. Repeat this step until the returned log page indicates that the operation has finished.

Observable Results:

1. Verify that the Device Self-test command returns status Invalid Field in Command.
2. Using the Log Page info returned in response to the second Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.

Case 6: **Test in Progress, Namespace Test Action = 00000000h, STC=1h and 1h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending an initial Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the

Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..

3. Configure the NVMe Host to send an initial Device Self-test command with Namespace Test Action set to 00000000h and STC set to 1h (Short device self-test operation). Wait for a Completion to be received with status "Success".
4. Determine that the initial Device Self-test command is in progress by sending a second Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 1 (Short device self-test operation in progress). If the Current Device Self-Test Operation field is set to 0, the Device Self-Test Operation is competing too quickly for the test to be performed and this test is not applicable.
5. Before the first Device Self-test operation completes, send a second Device Self-test command with Namespace Test Action set to 00000000h and STC set to 1h (Short device self-test operation). Wait for a Completion to be received.
6. Send a third Get Log Page command to LID 06h (Device Self-test). Repeat until the current Device Self-Test operation field is set to 0 in the Log Page that is returned.

Observable Results:

1. Using the Log Page info returned in response to the second Get Log Page command, which followed the initial Device Self-test command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 1h when the operation was processing.
2. Verify that the completion for the initial Device Self-test command indicates status "Success".
3. Verify that the completion for the second attempted Device Self-test command indicates status "Device Self-test in Progress".
4. Using the Log Page info returned in response to the third or later Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
5. Verify that all received responses have all Reserved fields set to 0.

Case 7: **Test in Progress, Namespace Test Action = 00000001h-FFFFFFFEh, STC=1h and 1h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send an initial Device Self-test command with Namespace Test Action set to a value between 00000001h and FFFFFFFEh and STC set to 1h (Short device self-test operation). Wait for a Completion to be received with status "Success".
4. Determine that the initial Device Self-test command is in progress by sending a second Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 1 (Short device self-test operation in progress). If the Current Device Self-Test Operation field is set to 0, the Device Self-Test Operation is competing too quickly for the test to be performed and this test is not applicable.
5. Before the first Device Self-test operation completes, send a second Device Self-test command with Namespace Test Action set to a value between 00000001h and FFFFFFFEh (use the same value as in the previous Device Self Test command) and STC set to 1h (Short device self-test operation). Wait for a Completion to be received.
6. Send a third Get Log Page command to LID 06h (Device Self-test). Repeat until the current Device Self-Test operation field is set to 0 in the Log Page that is returned.

Observable Results:

1. Using the Log Page info returned in response to the second Get Log Page command, which followed the initial Device Self-test command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 1h when the operation was processing.
2. Verify that the completion for the initial Device Self-test command indicates status "Success".
3. Verify that the completion for the second attempted Device Self-test command indicates status "Device Self-test in Progress".
4. Using the Log Page info returned in response to the third or later Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
5. Verify that all received responses have all Reserved fields set to 0.

Case 8: **Test in Progress, Namespace Test Action = FFFFFFFFh, STC=1h (FYI, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send an initial Device Self-test command with Namespace Test Action set to FFFFFFFFh and STC set to 1h (Short device self-test operation). Wait for a Completion to be received with status "Success".
4. Determine that the initial Device Self-test command is in progress by sending a second Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 1 (Short device self-test operation in progress). If the Current Device Self-Test Operation field is set to 0, the Device Self-Test Operation is competing too quickly for the test to be performed and this test is not applicable.
5. Before the first Device Self-test operation completes, send a second Device Self-test command with Namespace Test Action set to FFFFFFFFh and STC set to 1h (Short device self-test operation). Wait for a Completion to be received.
6. Send a third Get Log Page command to LID 06h (Device Self-test). Repeat until the current Device Self-Test operation field is set to 0 in the Log Page that is returned.

Observable Results:

1. Using the Log Page info returned in response to the second Get Log Page command, which followed the initial Device Self-test command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 1h when the operation was processing.
2. Verify that the completion for the initial Device Self-test command indicates status "Success".
3. Verify that the completion for the second attempted Device Self-test command indicates status "Device Self-test in Progress".
4. Using the Log Page info returned in response to the third or later Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
5. Verify that all received responses have all Reserved fields set to 0.

Possible Problems: None known.

Test 1.11 – Device Self-test Extended Operation (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly perform a Device Self-test Extended Operation.

References:

[1] NVMe Specification 5.8

Resource Requirements:

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

Last Modification: December 14, 2017

Discussion: The Device Self-test command is used to start a device self-test operation or abort a device self-test Operation. The device self-test operation is performed by the controller that the Device Self-test command was submitted to.

Test Setup: See Appendix A.

Case 1: **Namespace Test Action = 00000000h, STC=2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send a Device Self-test command with Namespace Test Action set to 00000000h and STC set to 2h (Extended device self-test operation).
4. Before the Device Self-test operation finishes, send a Get Log Page command to LID 06h (Device Self-test). Note that since the Device Self Test operation can be performed in the background, a command completion may be received before the Device Self Test operation finishes. Repeat this step until the returned log page indicates that the operation has finished.
5.
6. Report to the user the time elapsed between sending of the Device Self-test command and the operation finishing.

Observable Results:

1. Using the Log Page info returned in response to the first Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 2h when the operation was processing.
2. Verify that the Device Self-test command completes successfully.
3. Using the Log Page info returned in response to the second Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
4. Verify that all received responses have all Reserved fields set to 0.

Case 2: **Namespace Test Action = 00000001h-FFFFFFFEh, STC=2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.

2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to all value between 00000001h and FFFFFFFEh that represent a valid Namespace on the DUT, with STC set to 2h (Extended device self-test operation).
4. Before the Device Self-test operation finishes, send a Get Log Page command to LID 06h (Device Self-test). Note that since the Device Self Test operation can be performed in the background, a command completion may be received before the Device Self Test operation finishes. Repeat this step until the returned log page indicates that the operation has finished.
5.
6. Report to the user the time elapsed between sending of the Device Self-test command and the operation finishing.

Observable Results:

1. Using the Log Page info returned in response to the first Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 2h when the operation was processing.
2. Verify that the Device Self-test command completes successfully.
3. Using the Log Page info returned in response to the second Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
4. Verify that all received responses have all Reserved fields set to 0.

Case 3: **Namespace Test Action = FFFFFFFFh, STC=2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to FFFFFFFFh, with STC set to 2h (Extended device self-test operation).
4. Before the Device Self-test operation finishes, send a Get Log Page command to LID 06h (Device Self-test). Note that since the Device Self Test operation can be performed in the background, a command completion may be received before the Device Self Test operation finishes. Repeat this step until the returned log page indicates that the operation has finished.
5.
6. Report to the user the time elapsed between sending of the Device Self-test command and the operation finishing.

Observable Results:

1. Using the Log Page info returned in response to the first Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 2h when the operation was processing.
2. Verify that the Device Self-test command completes successfully.
3. Using the Log Page info returned in response to the second Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
4. Verify that all received responses have all Reserved fields set to 0.

Case 4: **Namespace Test Action = Invalid Namespace, STC=2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to an Invalid Namespace, with STC set to 2h (Extended device self-test operation).
4. Send a Get Log Page command to LID 06h (Device Self-test). Note that since the Device Self Test operation can be performed in the background, a command completion may be received before the Device Self Test operation finishes. Repeat this step until the returned log page indicates that the operation has finished.

Observable Results:

1. Verify that the Device Self-test command returns status Invalid Namespace or Format in Command.
2. Using the Log Page info returned in response to the second Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.

Case 5: **Namespace Test Action = Inactive Namespace, STC=2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Configure the Testing Station acting as a host to Check Bit 3 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports Namespace Management. If Bit 3 is not set to 1, this test is not applicable. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0.
3. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to an Inactive Namespace, with STC set to 2h (Extended device self-test operation).

Send a Get Log Page command to LID 06h (Device Self-test). Note that since the Device Self Test operation can be performed in the background, a command completion may be received before the Device Self Test operation finishes. Repeat this step until the returned log page indicates that the operation has finished.

Observable Results:

1. Verify that the Device Self-test command returns status Invalid Field in Command.
2. Using the Log Page info returned in response to the second Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.

Case 6: **Test in Progress, Namespace Test Action = 00000000h, STC=2h and 2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current

Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..

3. Configure the NVMe Host to send an initial Device Self-test command with Namespace Test Action set to 00000000h and STC set to 2h (Extended device self-test operation). Wait for a Completion to be received with status "Success".
4. Determine that the initial Device Self-test command is in progress by sending a second Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 2 (Extended device self-test operation in progress). If the Current Device Self-Test Operation field is set to 0, the Device Self-Test Operation is competing too quickly for the test to be performed and this test is not applicable.
5. Before the first Device Self-test operation completes, send a second Device Self-test command with Namespace Test Action set to 00000000h and STC set to 2h (Extended device self-test operation). Wait for a Completion to be received.
6. Send a third Get Log Page command to LID 06h (Device Self-test). Repeat until the current Device Self-Test operation field is set to 0 in the Log Page that is returned.

Observable Results:

1. Using the Log Page info returned in response to the second Get Log Page command, which followed the initial Device Self-test command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 1h when the operation was processing.
2. Verify that the completion for the initial Device Self-test command indicates status "Success".
3. Verify that the completion for the second attempted Device Self-test command indicates status "Device Self-test in Progress".
4. Using the Log Page info returned in response to the third or later Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
5. Verify that all received responses have all Reserved fields set to 0.

Case 7: **Test in Progress, Namespace Test Action = 00000001h-FFFFFFFEh, STC=2h and 2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send an initial Device Self-test command with Namespace Test Action set to a value between 00000001h and FFFFFFFEh and STC set to 2h (Extended device self-test operation). Wait for a Completion to be received with status "Success".
4. Determine that the initial Device Self-test command is in progress by sending a second Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 2 (Extended device self-test operation in progress). If the Current Device Self-Test Operation field is set to 0, the Device Self-Test Operation is competing too quickly for the test to be performed and this test is not applicable.
5. Before the first Device Self-test operation completes, send a second Device Self-test command with Namespace Test Action set to a value between 00000001h and FFFFFFFEh (use the same value as in the previous Device Self Test command) and STC set to 2h (Extended device self-test operation). Wait for a Completion to be received.
6. Send a third Get Log Page command to LID 06h (Device Self-test). Repeat until the current Device Self-Test operation field is set to 0 in the Log Page that is returned.

Observable Results:

1. Using the Log Page info returned in response to the second Get Log Page command, which followed the initial Device Self-test command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 1h when the operation was processing.
2. Verify that the completion for the initial Device Self-test command indicates status "Success".
3. Verify that the completion for the second attempted Device Self-test command indicates status "Device Self-test in Progress".
4. Using the Log Page info returned in response to the third or later Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
5. Verify that all received responses have all Reserved fields set to 0.

Case 8: **Test in Progress, Namespace Test Action = FFFFFFFFh, STC=2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host to send an initial Device Self-test command with Namespace Test Action set to FFFFFFFFh and STC set to 2h (Extended device self-test operation). Wait for a Completion to be received with status "Success".
4. Determine that the initial Device Self-test command is in progress by sending a second Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 2 (Extended device self-test operation in progress). If the Current Device Self-Test Operation field is set to 0, the Device Self-Test Operation is competing too quickly for the test to be performed and this test is not applicable.
5. Before the first Device Self-test operation completes, send a second Device Self-test command with Namespace Test Action set to FFFFFFFFh and STC set to 2h (Extended device self-test operation). Wait for a Completion to be received.
6. Send a third Get Log Page command to LID 06h (Device Self-test). Repeat until the current Device Self-Test operation field is set to 0 in the Log Page that is returned.

Observable Results:

1. Using the Log Page info returned in response to the second Get Log Page command, which followed the initial Device Self-test command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 1h when the operation was processing.
2. Verify that the completion for the initial Device Self-test command indicates status "Success".
3. Verify that the completion for the second attempted Device Self-test command indicates status "Device Self-test in Progress".
4. Using the Log Page info returned in response to the third or later Get Log Page command, verify that the NVMe Controller set the Current Device Self-test Status field in the Device Self-test Log to 0h when the operation was completed.
5. Verify that all received responses have all Reserved fields set to 0.

Possible Problems: None known.

Test 1.12 – Abort Device Self-test Short Operation (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly perform an abort on a Device Self-test Short Operation.

References:

[1] NVMe Specification 5.8, 8.11, TP 4022

Resource Requirements:

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

Last Modification: May 4, 2020

Discussion: The Device Self-test command is used to start a device self-test operation or abort a device self-test Operation. The device self-test operation is performed by the controller that the Device Self-test command was submitted to.

Test Setup: See Appendix A.

Case 1: **Namespace Test Action = 00000000h, STC=1h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send a Device Self-test command with Namespace Test Action set to 00000000h and STC set to 1h (Short device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Device Self-test command with Namespace Test Action set to 00000000h and STC set to Fh (Abort device self-test), to the same controller as in the previous step.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=1h, completes with status 1h, Aborted by a Device Self Test Command.
2. Verify that the second Device Self-test command, with STC=Fh, completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.
4. Verify that the second received Device Self-test Log has the Current Device Self-test Status field set to 0h.
5. Verify that all received responses have all Reserved fields set to 0.

Case 2: **Test Action = 00000001h-FFFFFFFEh, STC=1h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.

2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to a value between 00000001h and FFFFFFFEh that represent a valid Namespace on the DUT, with STC set to 1h (Short device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Device Self-test command with Namespace Test Action set the same value as the previous step, and STC set to Fh (Abort device self-test), to the same controller as in the previous test.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=1h, completes with status 1h, Aborted by a Device Self Test Command.
2. Verify that the second Device Self-test command, with STC=Fh, completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.
4. Verify that the second received Device Self-test Log has the Current Device Self-test Status field set to 0h.
5. Verify that all received responses have all Reserved fields set to 0.

Case 3: **Test Action = FFFFFFFFh, STC=1h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to a value of FFFFFFFFh, with STC set to 1h (Short device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Device Self-test command with Namespace Test Action set the same value as the previous step, and STC set to Fh (Abort device self-test), to the same controller as in the previous test.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=1h, completes with status 1h, Aborted by a Device Self Test Command.
2. Verify that the second Device Self-test command, with STC=Fh, completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.
4. Verify that the second received Device Self-test Log has the Current Device Self-test Status field set to 0h.
5. Verify that all received responses have all Reserved fields set to 0.

Case 4: **DST with specific NSID Aborted by Format NVM Command with specific NSID (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to a valid Namespace ID value other than NSID=FFFFFFFFh, with STC set to 1h (Short device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Format NVM command with the same Namepace ID as in the previous step.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=1h, completes with status 4h aborted by a Format NVM command.
2. Verify that the Format NVM command completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.
4. Verify that all received responses have all Reserved fields set to 0.

Case 5: **Aborted by Controller Level Reset (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to a valid Namespace ID value, with STC set to 1h (Short device self-test operation).
5. Configure the NVMe Host to perform a Controller Level Reset.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=1h, completes with status 2h aborted by a Controller Level Reset.
2. Verify that the Controller Level Reset completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.
4. Verify that the second received Device Self-test Log has the Current Device Self-test Status field set to 0h.

5. Verify that all received responses have all Reserved fields set to 0.

Case 6: **Aborted by Sanitize Operation (FYI, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Check SANICAP field in the Identify Controller Data Structure. If SANICAP is set to 0, this test is not applicable.
3. Check the VER field. This test is only applicable if the DUT claims to support NVMe v1.4 or higher.
4. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
5. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
6. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to a valid Namespace ID value, with STC set to 1h (Short device self-test operation).
7. Configure the Testing Station to issue a Sanitize Command for a supported operation type as indicated as supported in the SANICAP field: Overwrite, Block Erase, and Crypto Erase. Repeat entire test procedure for each support operation type.
8. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=1h, completes with status "Aborted due to Sanitize (9h)" indicated in the Self-test Result Data Structure returned in response to the Get Log Page request.

Case 7: **DST with specific NSID Aborted by Format NVM Command with NSID = FFFFFFFFh (FYI, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to a valid Namespace ID value other than NSID=FFFFFFFFh, with STC set to 1h (Short device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Format NVM command with NSID=FFFFFFFFh.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=1h, completes with status 4h aborted by Format NVM command.
2. Verify that the Format NVM command completes successfully.

3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.
4. Verify that all received responses have all Reserved fields set to 0.

Case 8: **DST NSID=FFFFFFFFh Aborted by Format NVM Command with NSID = FFFFFFFFh (FYI, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test commands with NSID=FFFFFFFFh, with STC set to 1h (Short device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Format NVM command with NSID=FFFFFFFFh.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=1h, completes with status 4h aborted by Format NVM command.
2. Verify that the Format NVM command completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.

Possible Problems: None known.

Test 1.13 – Abort Device Self-test Extended Operation (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly perform an abort on a Device Self-test Extended Operation.

References:

[1] NVMe Specification 5.8, 8.11, TP 4022

Resource Requirements:

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

Last Modification: May 4, 2020

Discussion: The Device Self-test command is used to start a device self-test operation or abort a device self-test Operation. The device self-test operation is performed by the controller that the Device Self-test command was submitted to.

Test Setup: See Appendix A.

Case 1: **Namespace Test Action = 00000000h, STC=2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send a Device Self-test command with Namespace Test Action set to 00000000h and STC set to 2h (Extended device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Device Self-test command with Namespace Test Action set to 00000000h and STC set to Fh (Abort device self-test), to the same controller as in the previous test.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=2h, completes with status 1h Aborted by a Device Self Test command..
2. Verify that the second Device Self-test command, with STC=Fh, completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.
4. Verify that the second received Device Self-test Log has the Current Device Self-test Status field set to 0h.
5. Verify that all received responses have all Reserved fields set to 0.

Case 2: **Test Action = 00000001h-FFFFFFFEh, STC=2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to a value between 00000001h and FFFFFFFEh that represent a valid Namespace on the DUT, with STC set to 2h (Extended device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Device Self-test command with Namespace Test Action set the same value as the previous step, and STC set to Fh (Abort device self-test), to the same controller as in the previous test.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=2h, completes with status 1h, Aborted by a Device Self Test Command..
2. Verify that the second Device Self-test command, with STC=Fh, completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.
4. Verify that the second received Device Self-test Log has the Current Device Self-test Status field set to 0h.
5. Verify that all received responses have all Reserved fields set to 0.

Case 3: **Test Action = FFFFFFFFh, STC=2h (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to a value of FFFFFFFFh, with STC set to 2h (Extended device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Device Self-test command with Namespace Test Action set the same value as the previous step, and STC set to Fh (Abort device self-test), to the same controller as in the previous test.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=2h, completes with status 1h, Aborted by a Device Self Test Command..
2. Verify that the second Device Self-test command, with STC=Fh, completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.

4. Verify that the second received Device Self-test Log has the Current Device Self-test Status field set to 0h.
5. Verify that all received responses have all Reserved fields set to 0.

Case 4: **DST with specific NSID Aborted by Format NVM Command with specific NSID (M, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0.
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to a valid Namespace ID value other than FFFFFFFFh, with STC set to 2h (Extended device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Format NVM command with the name Namepace ID as in the previous step.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=2h, completes with status 4h, to indicate that the Device Self Test Operation was aborted by a Format NVM command..
2. Verify that the Format NVM command completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.
4. Verify that all received responses have all Reserved fields set to 0.

Case 5: **Aborted by Sanitize Operation (FYI, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Check SANICAP field in the Identify Controller Data Structure. If SANICAP is set to 0, this test is not applicable.
3. Check the VER field. This test is only applicable if the DUT claims to support NVMe v1.4 or higher.
4. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
5. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
6. Configure the NVMe Host to send Device Self-test commands with Namespace Test Action set to a valid Namespace ID value, with STC set to 2h (Extended device self-test operation).
7. Configure the Testing Station to issue a Sanitize Command for a supported operation type as indicated as supported in the SANICAP field: Overwrite, Block Erase, and Crypto Erase. Repeat entire test procedure for each support operation type.
8. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test command, with STC=2h, completes with status "Aborted due to Sanitize (9h)" indicated in the Self-test Result Data Structure returned in response to the Get Log Page request..

Case 6: **DST with specific NSID Aborted by Format NVM Command with NSID = FFFFFFFFh (FYI, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test Extended Operation commands with Namespace Test Action set to a valid Namespace ID value other than NSID=FFFFFFFFh, with STC set to 2h (Extended device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Format NVM command with NSID=FFFFFFFFh.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test Extended Operation command, with STC=2h, completes with status 4h, to indicate that the Device Self Test Operation was aborted by a Format NVM command..
2. Verify that the Format NVM command completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.
4. Verify that all received responses have all Reserved fields set to 0.

Case 7: **DST NSID=FFFFFFFFh Aborted by Format NVM Command with NSID = FFFFFFFFh (FYI, OF-FYI)**

Test Procedure:

1. Configure the Testing Station acting as a host to Check Bit 4 of the OACS field in the Identify Controller Data Structure to determine if the DUT supports the Device Self-test command. If Bit 4 is not set to 1, this test is not applicable.
2. Ensure that no Device Self-test operation is currently in progress by sending Get Log Page with LID=06h to the DUT. Check that Current Device Self-Test Operation field in the response is set to 0. If the Current Device Self-Test Operation field is not set to 0, wait for the Device Self-Test Operation to complete, and send the Get Log Page with LID=06h again. Repeat until the Current Device Self-Test Operation field is 0..
3. Configure the NVMe Host send a Get Log Page command to LID 06h (Device Self-test).
4. Configure the NVMe Host to send Device Self-test commands with NSID=FFFFFFFFh, with STC set to 2h (Extended device self-test operation).
5. Before the Device Self-test operation completes in the background, send a Format NVM command with NSID=FFFFFFFFh.
6. Once a command completion is received for the Device Self-test command, send a Get Log Page command to LID 06h (Device Self-test). Repeat until the log Page indicates that the DST operation is no longer in progress.

Observable Results:

1. Verify that the first Device Self-test Extended Operation command, with STC=2h, completes with status 4h, to indicate that the Device Self Test Operation was aborted by a Format NVM command..
2. Verify that the Format NVM command completes successfully.
3. Compare the Device Self Test Log received in steps 3 and 6 and verify that that the second received Device Self Test Log contains a new log entry in the Newest Self-test Results Data Structure in the Device Self-test Log.

Possible Problems: None known.

Test 1.14 – NVMe-MI Send/Receive (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller can properly perform an NVMe-MI Send and Receive operation.

References:

[1] NVMe Specification 5.17, 5.18
[2] NVMe-MI Specification 5.5

Resource Requirements:

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

Last Modification: February 5, 2018

Discussion: The NVMe-MI Send command is used to transfer an NVMe-MI Request Message to the controller.

NVMe-MI Receive command transfers an NVMe-MI Response Message from the controller to the host that corresponds to an NVMe-MI Request Message that was previously submitted to the controlle.

Refer to the NVM Express Management Interface (NVMe-MI) specification for the format and servicing of the NVMe-MI Request and Response Messages.

Test Setup: See Appendix A.

Case 1: **Request and Response (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 6 of the OACS field in the Identify Controller Data Structure. If Bit 6 is not set to 1, this test is not applicable.
2. Configure the Testing Station to issue an NVMe-MI Send Command containing a pointer to a properly formatted NVMe-MI Request for Read NVMe-MI Data Structure with DTYP = 00h – NVM Subsystem Information.
3. Configure the Testing Station to issue an NVMe-MI Receive Command to receive the NVMe-MI Response associated with the NVMe-MI Request in the previous step.

Observable Results:

1. Verify that for both the NVMe-MI Send and Receive commands that a command completion queue entry is posted to the Admin Completion Queue.
2. Verify that the Controller sent a properly formatted NVMe-MI Response to the NVMe-MI Request for Read NVMe-MI Data Structure with DTYP = 00h – NVM Subsystem Information, with a properly formatted NVM Subsystem Information data structure.

Possible Problems: None known.

Test 1.15 – Directive Receive Identify (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller can properly perform a Directives Receive operation.

References:

[1] NVMe Specification 5.9, 5.10, 9

Resource Requirements:

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

Last Modification: November 26, 2018

Discussion: Directives is a mechanism to enable host and NVM subsystem or controller information exchange. The Directive Receive command is used to transfer data related to a specific Directive Type from the controller to the host. The Directive Send command is used to transfer data related to a specific Directive Type from the host to the controller.

Test Setup: See Appendix A.

Case 1: **Valid Receive (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 5 of the OACS field in the Identify Controller Data Structure. If Bit 5 is not set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Directive Receive Command with Directive Type set to Identify (00h) and the Directive Operation Value set to Return parameters (01h), and a valid NSID value.

Observable Results:

1. Verify that the controller sent a Return Parameters Data Structure indicating that the identify Directive is support and enabled.

Case 2: **Receive with NSID=FFFFFFFFh (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 5 of the OACS field in the Identify Controller Data Structure. If Bit 5 is not set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Directive Receive Command with Directive Type set to Identify (00h) and the Directive Operation Value set to Return parameters (01h), and a NSID value of FFFFFFFFh.

Observable Results:

1. Verify that the Directive Receive command completes with status Invalid Field in Command.

Possible Problems: None known.

Test 1.16 – Directive Send Enable Directive (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller can properly perform a Directives Send operation.

References:

[1] NVMe Specification 5.9, 5.10, 9

Resource Requirements:

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

Last Modification: April 2, 2019

Discussion: Directives is a mechanism to enable host and NVM subsystem or controller information exchange. The Directive Receive command is used to transfer data related to a specific Directive Type from the controller to the host. The Directive Send command is used to transfer data related to a specific Directive Type from the host to the controller.

Test Setup: See Appendix A.

Case 1: **Valid Send (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 5 of the OACS field in the Identify Controller Data Structure. If Bit 5 is not set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Directive Receive Command with Directive Type set to Identify (00h) and the Directive Operation Value set to Return parameters (01h), and a valid NSID value.
3. Check the Return Parameters Data Structure if the Streams Directive is supported. If the Streams Directive is not supported this test is not applicable.
4. If the Streams Directive is supported, and not enabled, the Testing Station should perform a Directive Send with operation Enable Directive (01h) and DTYPE set to Streams (01h), and ENDIR set to 1.
5. If the Streams Directive is supported, and enabled, the Testing Station should perform a Directive Send with operation Enable Directive (01h) and DTYPE set to Streams (01h), and ENDIR set to 0. Next, the Testing Station should perform a Directive Send with operation Enable Directive (01h) and DTYPE set to Streams (01h), and ENDIR set to 1.

Observable Results:

1. Verify that the controller sent a Return Parameters Data Structure indicating that the identify Directive is support and enabled.
2. Verify that each of the Directives Send commands completes with status “Success”.

Case 2: **Send to Enable Identify (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 5 of the OACS field in the Identify Controller Data Structure. If Bit 5 is not set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Directive Receive Command with Directive Type set to Identify (00h) and the Directive Operation Value set to Return parameters (01h), and a valid NSID value.
3. Check the Return Parameters Data Structure if the Streams Directive is supported. If the Streams Directive is not supported this test is not applicable.
4. The Testing Station should perform a Directive Send with operation Enable Directive (01h) and DTYPE set to Identify (00h), and ENDIR set to 1.

Observable Results:

1. Verify that the controller sent a Return Parameters Data Structure indicating that the identify Directive is support and enabled.
2. Verify that each of the Directives Send commands completes with status “Success”.

Case 3: **Send to Enable Unsupported Directive (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 5 of the OACS field in the Identify Controller Data Structure. If Bit 5 is not set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Directive Receive Command with Directive Type set to Identify (00h) and the Directive Operation Value set to Return parameters (01h), and a valid NSID value.
3. Check the Return Parameters Data Structure if the Streams Directive is supported. If the Streams Directive is supported this test is not applicable.
4. The Testing Station should perform a Directive Send with operation Enable Directive (01h) and DTYPE set to Streams (01h), and ENDIR set to 1.

Observable Results:

1. Verify that the Directive Send command completes with status Invalid Field in Command.

Case 4: **Shared Stream Writes (M, OF-FYI)**

Test Procedure:

1. Check Bit 5 of the OACS field in the Identify Controller Data Structure. If Bit 5 is not set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Directive Receive Command with Directive Type set to Identify (00h) and the Directive Operation Value set to Return parameters (01h), and a valid NSID value.
3. Check the Return Parameters Data Structure. If the Streams Directive is supported. If the Streams Directive is supported this test is not applicable.
4. The Testing Station should perform a Directive Send with operation Enable Directive (01h) and DTYPE set to Streams (01h), and ENDIR set to 1.
5. Issue an Allocate Resources operation to allocate stream resources to the namespace from 2 different Host IDs.
6. Issue multiple write commands to the namespace from 2 different Host IDs using the same stream identifier (Shared Stream Writes). Each Host will write a unique data pattern.
7. Perform a READ operation to the LBAs written in the previous step.

Observable Results:

1. Verify that the Shared Stream Write operations completed successfully and that the expected data pattern was read back.

Case 5: **Directive Send Release Resources updates NSSA (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 5 of the OACS field in the Identify Controller Data Structure. If Bit 5 is not set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Directive Receive Command with Directive Type set to Identify (00h) and the Directive Operation Value set to Return parameters (01h), and a valid NSID value.
3. Check the Return Parameters Data Structure. If the Streams Directive is supported. If the Streams Directive is supported this test is not applicable.
4. The Testing Station should perform a Directive Send with operation Enable Directive (01h) and DTYPE set to Streams (01h), and ENDIR set to 1.
5. Perform a Direct Receive Operation of Return Parameters, records the NSSA value.

6. Issue an Allocate Resources operation (03h) to allocate stream resources to the namespace from a single Host IDs with an NSR value of 1.
7. Perform a Direct Receive Operation of Return Parameters, records the NSSA value.
8. Issue an Release Resources operation (02h) to release stream resources from the namespace.
9. Perform a Direct Receive Operation of Return Parameters, records the NSSA value.

Observable Results:
1. Verify that the NSSA value reported in Step 7 was 1 less than the NSSA value reported in Step 5.
2. Verify that the NSSA value reported in Step 9 was equal to the NSSA value reported in Step 5.

Case 6: **Namespace Deletion updates NSSA (FYI, OF-FYI)**

Test Procedure:
1. Check Bit 5 of the OACS field in the Identify Controller Data Structure. If Bit 5 is not set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Directive Receive Command with Directive Type set to Identify (00h) and the Directive Operation Value set to Return parameters (01h), and a valid NSID value.
3. Check the Return Parameters Data Structure. If the Streams Directive is supported. If the Streams Directive is supported this test is not applicable.
4. The Testing Station should perform a Directive Send with operation Enable Directive (01h) and DTYPE set to Streams (01h), and ENDIR set to 1.
5. Perform a Direct Receive Operation of Return Parameters, records the NSSA value.
6. Issue an Allocate Resources operation (03h) to allocate stream resources to the namespace from a single Host IDs with an NSR value of 1.
7. Perform a Direct Receive Operation of Return Parameters, records the NSSA value.
8. Delete the namespace resources were allocated to.
9. Perform a Direct Receive Operation of Return Parameters, records the NSSA value.

Observable Results:
1. Verify that the NSSA value reported in Step 7 was 1 less than the NSSA value reported in Step 5.
2. Verify that the NSSA value reported in Step 9 was equal to the NSSA value reported in Step 5.

Case 7: **Format NVM to Namespace updates NSSA (FYI, OF-FYI)**

Test Procedure:
1. Check Bit 5 of the OACS field in the Identify Controller Data Structure. If Bit 5 is not set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Directive Receive Command with Directive Type set to Identify (00h) and the Directive Operation Value set to Return parameters (01h), and a valid NSID value.
3. Check the Return Parameters Data Structure. If the Streams Directive is supported. If the Streams Directive is supported this test is not applicable.
4. The Testing Station should perform a Directive Send with operation Enable Directive (01h) and DTYPE set to Streams (01h), and ENDIR set to 1.
5. Perform a Direct Receive Operation of Return Parameters, records the NSSA value.
6. Issue an Allocate Resources operation (03h) to allocate stream resources to the namespace from a single Host IDs with an NSR value of 1.
7. Perform a Direct Receive Operation of Return Parameters, records the NSSA value.
8. Perform a Format NVM command on the namespace resources were allocated to.
9. Perform a Direct Receive Operation of Return Parameters, records the NSSA value.

Observable Results:
1. Verify that the NSSA value reported in Step 7 was 1 less than the NSSA value reported in Step 5.
2. Verify that the NSSA value reported in Step 9 was equal to the NSSA value reported in Step 5.

Possible Problems: None known.

Test 1.17 – Sanitize Command (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller can properly perform a Sanitize operation.

References:

[1] NVMe Specification 5.24, 8.15, NVMe v1.3 ECN 005

Resource Requirements:

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

Last Modification: December 1, 2019

Discussion: The Sanitize command is used to start a sanitize operation or to recover from a previously failed sanitize operation. The sanitize operation types that may be supported are Block Erase, Crypto Erase, and Overwrite. All sanitize operations are processed in the background (i.e., completion of the Sanitize command does not indicate completion of the sanitize operation).

Test Setup: See Appendix A.

Case 1: **Sanitize Supported (M, OF-FYI)**

Test Procedure:

1. Check SANICAP field in the Identify Controller Data Structure. If SANICAP is set to 0, this test is not applicable.
2. Configure the Testing Station to issue a Sanitize Command, for each of the operation types indicated as supported in the SANICAP field: Overwrite, Block Erase, and Crypto Erase. Each Sanitize command should be followed by a Get Log Page command for the Sanitize Status Log Page.

Observable Results:

1. Verify that each of the supported Sanitize operation commands completes with status “Success”, and each of the unsupported Sanitize operation commands completes with status “Invalid Field in Command”.
2. Verify that the Sanitize Status Log Page is updated after each Sanitize operation that completes with status “success”.

Case 2: **Sanitize Not Supported (M, OF-FYI)**

Test Procedure:

1. Check SANICAP field in the Identify Controller Data Structure.
2. Configure the Testing Station to issue a Sanitize Command, repeat for each operation: Overwrite, Block Erase, and Crypto Erase.

Observable Results:

1. Verify that each of the unsupported Sanitize operations completes with status “Invalid Field in Command” and that the corresponding bit in the SANICAP field is set to 0.

Case 3: **Sanitize Config FID Not Savable (FYI, OF-FYI)**

Test Procedure:

1. Check SANICAP field in the Identify Controller Data Structure. If SANICAP is set to 0, this test is not applicable.
2. Configure the Testing Station to issue a Get Feature Command for FID 17h Sanitize Config with SEL=011b (Supported Capabilities).

Observable Results:

1. Verify that if Dword 0 bit 0 of the completion entry of the Get Features command with SEL=011b (i.e., Supported Capabilities) for this Feature Identifier is cleared to '0', then the default value of the NODRM attribute shall be cleared to '0'.

Case 4: **Sanitize Config NDI=1, NODRM=1 (FYI, OF-FYI)**

Test Procedure:

1. Check SANICAP field in the Identify Controller Data Structure. If the No Deallocate Inhibited bit (NDI)is set to 0, this test is not applicable.
2. Configure the Testing Station to issue a Set Feature Command for FID 17h Sanitize Config to set NODRM =1.
3. Configure the Testing Station to issue a Get Feature Command for FID 17h Sanitize Config to verify that NODRM =1.
4. Perform a Sanitize Command with the No Deallocate After Sanitize Bit set to 1.
5. Perform a Get Log Page for the Sanitize Status Log Page, LID=81h.

Observable Results:

1. Verify that the Sanitize operation is completed successfully.
2. Verify that bits 2:0 of the Sanitize Status field in the Sanitize Status log page are set to 100b.

Case 5: **Sanitize Config NDI=1, NODRM=0 (FYI, OF-FYI)**

Test Procedure:

1. Check SANICAP field in the Identify Controller Data Structure. If the No Deallocate Inhibited bit (NDI)is set to 0, this test is not applicable.
2. Configure the Testing Station to issue a Set Feature Command for FID 17h Sanitize Config to set NODRM =0.
3. Configure the Testing Station to issue a Get Feature Command for FID 17h Sanitize Config to verify that NODRM =0.
4. Perform a Sanitize Command with the No Deallocate After Sanitize Bit set to 1.
5. Perform a Get Log Page for the Sanitize Status Log Page, LID=81h.

Observable Results:

1. Verify that the Sanitize operation is aborted with status 'Invalid Field in Command'.

Case 6: **Sanitize In Progress SPROG (FYI, OF-FYI)**

Test Procedure:

1. Check SANICAP field in the Identify Controller Data Structure. If the SANICAP field set to 0, this test is not applicable.
2. Perform a Sanitize Command, before command completion arrives for the Sanitize Command, issue a Get Log Page for the Sanitize Status Log Page, LID=81h.

Observable Results:

1. Verify that the Sanitize Status Log Page has SSTAT set to 010b, and SPROG is not equal to FFFFh.

Case 7: **Sanitize Not In Progress SPROG (FYI, OF-FYI)**

Test Procedure:

1. Check SANICAP field in the Identify Controller Data Structure. If the SANICAP field set to 0, this test is not applicable.
2. Perform a Sanitize Command, wait for command completion.
3. Perform a Get Log Page for the Sanitize Status Log Page, LID=81h.

Observable Results:

1. Verify that the Sanitize Status Log Page has SSTAT set to 001b, and SPROG=FFFFh.

Possible Problems: None known.

Test 1.18 – Virtualization Management Command (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller properly supports the Virtualization Management command if supported.

References:

[1] NVMe Specification 5.22, 8.5

Resource Requirements:

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

Last Modification: December 1, 2019

Discussion: The Virtualization Management command is supported by primary controllers that support the Virtualization Enhancements capability. This command is used for Modifying Flexible Resource allocation for the primary controller, Assigning Flexible Resources for secondary controllers, and Setting the Online and Offline state for secondary controllers.

Test Setup: See Appendix A.

Case 1: **Valid Virtualization Management Command, RT=000b (FYI, OF-FYI)**

Test Procedure:

1. Check OACS field Bit 7 in the Identify Controller Data Structure to determine if the DUT supports the Virtualization Management command. If the DUT does not support the Virtualization Management command, this test is not applicable.
2. Perform an Identify command for CSN=14h and record the Primary Controller Capabilities Structure.
3. Configure the Testing Station to issue a Virtualization Management Command with Resource Type = VQ Resources (000b), ACT=1h (Primary Controller Flexible Allocation), valid CTRLID, valid NR (Number of Controller Resources).

Observable Results:

1. If Bit 0 (VQ Resources) of the CRT field in the Primary Controller Capabilities Structure is set to 1, then verify that the Virtualization Management commands completes with status Success. or Invalid Resource Identifier.
2. If Bit 0 (VQ Resources) of the CRT field in the Primary Controller Capabilities Structure is set to 0, then verify that the Virtualization Management commands completes with status Invalid Resource Identifier.

Case 2: **Valid Virtualization Management Command, RT=001b (FYI, OF-FYI)**

Test Procedure:

1. Check OACS field Bit 7 in the Identify Controller Data Structure to determine if the DUT supports the Virtualization Management command. If the DUT does not support the Virtualization Management command, this test is not applicable.
2. Perform an Identify command for CSN=14h and record the Primary Controller Capabilities Structure.
3. Configure the Testing Station to issue a Virtualization Management Command with Resource Type = VI Resources (001b), ACT=1h (Primary Controller Flexible Allocation), valid CTRLID, valid NR (Number of Controller Resources).

Observable Results:

1. If Bit 1 (VI Resources) of the CRT field in the Primary Controller Capabilities Structure is set to 1, then verify that the Virtualization Management commands completes with status Success.

2. If Bit 1 (VI Resources) of the CRT field in the Primary Controller Capabilities Structure is set to 0, then verify that the Virtualization Management commands completes with status Invalid Resource Identifier.

Case 3: **Invalid CNTLID (FYI, OF-FYI)**

Test Procedure:

1. Check OACS field Bit 7 in the Identify Controller Data Structure to determine if the DUT supports the Virtualization Management command. If the DUT does not support the Virtualization Management command, this test is not applicable.
2. Perform an Identify command for CSN=14h and record the Primary Controller Capabilities Structure.
3. Configure the Testing Station to issue a Virtualization Management Command with Resource Type set to a supported value (based on the contents of the Primary Controller Capabilities Structure), ACT=1h (Primary Controller Flexible Allocation), valid NR (Number of Controller Resource, and an invalid CNTLID.

Observable Results:

1. Verify that the Virtualization Management commands completes with status Invalid Resource Identifier.

Possible Problems: None known.

Test 1.19 – Security Receive (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller can properly perform a Security Receive operation.

References:

[1] NVMe Specification 5.25

Resource Requirements:

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

Last Modification: May 5, 2020

Discussion: The Security Receive command transfers the status and data result of one or more Security Send commands that were previously submitted to the controller.

Test Setup: See Appendix A.

Case 1: **SECP = 00h (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 0 of the OACS field in the Identify Controller Data Structure. If Bit 0 is set to 0, this test is not applicable.
2. Configure the Testing Station to issue a Security Receive Command with SECP=00h.

Observable Results:

1. Verify that the controller returned a list of supported security protocols.

Case 2: **SECP = Unsupported Value (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 0 of the OACS field in the Identify Controller Data Structure. If Bit 0 is set to 0, this test is not applicable.
2. Configure the Testing Station to issue a Security Receive Command with SECP=00h.
3. Configure the Testing Station to issue a Security Receive Command with SECP= any value not included in the list of supported security protocols returned in response to the previous Security Receive command.

Observable Results:

1. Verify that the second Security Receive command aborted with status Invalid Field in Command.

Possible Problems: None known.

Test 1.20 – Security Send (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller can properly perform a Security Send operation.

References:

[1] NVMe Specification 5.26

Resource Requirements:

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

Last Modification: May 5, 2020

Discussion: The Security Send command is used to transfer security protocol data to the controller. The data structure transferred to the controller as part of this command contains security protocol specific commands to be performed by the controller.

Test Setup: See Appendix A.

Case 1: **SECP = Unsupported Value (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 0 of the OACS field in the Identify Controller Data Structure. If Bit 0 is set to 0, this test is not applicable.
2. Configure the Testing Station to issue a Security Receive Command with SECP=00h.
3. Configure the Testing Station to issue a Security Send Command with SECP= any value not included in the list of supported security protocols returned in response to the previous Security Receive command.

Observable Results:

1. Verify that the second Security Send command aborted with status Invalid Field in Command.

Possible Problems: None known.

Group 2: NVM Command Set

Overview:

This section describes a method for performing conformance verification for NVMe products implementing the NVM Command Set.

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 – Compare Command (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly execute the Compare command.

References:

[1] NVMe Specification 6.6, 8.3.1.4, 8.3.1.4, NVMe v1.3 ECN 001

Resource Requirements:

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

Last Modification: April 8, 2019

Discussion: The Compare command reads the logical blocks specified by the command from the medium and compares the data read to a comparison data buffer transferred as part of the command. If the data read from the controller and the comparison data buffer are equivalent with no miscompares, then the command completes successfully. If there is any miscompare, the command completes with an error of Compare Failure. If metadata is provided, then a comparison is also performed for the metadata.

The command uses Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 14, and Command Dword 15. If the command uses PRPs for the data transfer, then the Metadata Pointer, PRP Entry 1, and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the Metadata SGL Segment Pointer and SGL Entry 1 fields are used. All other command specific fields are reserved.

Test Setup: See Appendix A.

Case 1: **Valid SLBA (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports the Compare command. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller in order to write a known sequence to the LBA 0000h.
3. Configure the NVMe Host to issue a Compare command comparing the known sequence to the LBA 0000h written to in Step 1.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the Compare command completes successfully.

Case 2: **SLBA Out of Range (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports the Compare command. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller in order to write a known sequence to the LBA 0000h.
4. Configure the NVMe Host to issue a Compare command to an SLBA which is out of range of the DUT, comparing the known sequence to the LBA 0000h written to in Step 1.

Observable Results:

1. Verify that the Compare command completes with status code LBA Out of Range (80h).

Case 3: **SLBA In Range, NLB Goes out of range (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports the Compare command. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller in order to write a known sequence to the LBA 0000h.
3. Configure the NVMe Host to issue a Compare command to the controller to an SLBA which is in range of the DUT, but an NLB value that will push the Compare Command past the size of the namespace, comparing the known sequence to the LBA 0000h written to in Step 1.

Observable Results:

1. Verify that the Compare command completes with status code LBA Out of Range (80h).

Case 4: **SLBA Out of Range, NLB > MDTS (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports the Compare command. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller in order to write a known sequence to the LBA 0000h.
3. Configure the NVMe Host to issue a Compare command to the controller comparing the known sequence to the LBA 0000h written to in Step 1, to an SLBA which out of range of the DUT, and an NLB which is greater than MDTS. If MDTS is set to zero (unlimited), then this test case is not applicable.

Observable Results:

1. Verify that the Compare command completes with an error status code. The DUT can report any error status code.

Case 5: **SLBA Out of Range, but Lower Dword = 00000000 (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports the Compare command. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller in order to write a known sequence to the LBA 0000h.
3. Configure the NVMe Host to issue a Compare command to the controller comparing the known sequence to the LBA 0000h written to in Step 1, with an SLBA of FFFFFFFF00000000h ,which, when read as a 64 bit value is out of range of the DUT, and an NLB which is less than MDTS.

Observable Results:

1. Verify that the Compare command completes with status code LBA Out of Range (80h).

Case 6: **Invalid Namespace ID (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports the Compare command. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller in order to write a known sequence to the LBA 0000h.
3. Configure the NVMe Host to issue a Compare command to the controller comparing the known sequence to the LBA 0000h written to in Step 1, specifying an Invalid NSID outside of the controller specified range defined by NN (Number of Namespaces). If NN is the maximum possible value (0xFFFFFFFF) this sub test cannot be performed.

Observable Results:

1. Verify that the Compare command completes with status code Invalid Namespace or Format (0Bh).

Case 7: **PRCHK Non-zero (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports the Compare command. If the command is not supported this test is not applicable.
2. Ensure that the namespace under test is formatted with end to end data protection. If End to End Data Protection is not supported then this test is not applicable.
3. Configure the NVMe Host to issue a Write command with PRACT set to 1, and PRCHK Bit 00 set to 1, to the controller in order to write a known sequence to the LBA 0000h.
4. Configure the NVMe Host to issue a Compare command to the controller with PRACT set to 0, and PRCHK Bit 00 set to 1, comparing the known sequence to the LBA 0000h written to in Step 3.
5. Repeat steps 3 and 4 with PRCHK Bit 01 set to 1, and again with PRCHK Bit 02 set to 1.

Observable Results:

1. Verify that in each case the Compare command completes successfully.

Case 8: **NSID=FFFFFFFFh (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports the Compare command. If the command is not supported this test is not applicable.
2. Check the NVMe specification version supported by the DUT. If the specification is not v1.4 or higher, this test is not applicable.
3. Configure the NVMe Host to issue a Write command to the controller in order to write a known sequence to the LBA 0000h.
4. Configure the NVMe Host to issue a Compare command with NSID=FFFFFFFFh comparing the known sequence to the LBA 0000h written to in Step 1.

Observable Results:

1. Verify that the Compare command did not complete successfully.

Possible Problems: None.

Test 2.2 – Dataset Management Command (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly execute the Dataset Management command.

References:

[1] NVMe Specification 6.7

Resource Requirements:

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

Last Modification: November 26, 2018

Discussion: The Dataset Management command is used by the host to indicate attributes for ranges of logical blocks. This includes attributes like frequency that data is read or written, access size, and other information that may be used to optimize performance and reliability. This command is advisory; a compliant controller may choose to take no action based on information provided.

The command uses Command Dword 10 and Command Dword 11 fields. If the command uses PRPs for the data transfer, then the PRP Entry 1 and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, the SGL Entry 1 field is used. All other command specific fields are reserved.

The data that the Dataset Management command provides is a list of ranges with context attributes. Each range consists of a starting LBA, a length of logical blocks that the range consists of and the context attributes to be applied to that range. The definition of the Dataset Management command Range field is specified in Table 5. The maximum case of 256 ranges is shown.

Table 5 – Dataset Management Command Range Definition

Range	Byte	Field
Range 0	03:00	Context Attributes
	07:04	Length in logical blocks
	15:08	Starting LBA
Range 1	19:16	Context Attributes
	23:20	Length in logical blocks
	31:24	Starting LBA
…		
Range 255	4083:4080	Context Attributes
	4087:4084	Length in logical blocks
	4095:4088	Starting LBA

Test Setup: See Appendix A.

Case 1: **Basic Operation (M, OF)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure or check for non-zero values in the DMRL, DMRSL, and DMSL fields to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue the Dataset Management command that indicates attributes for ranges of logical blocks.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

Case 2: **Deallocate (M, OF)**

The Dataset Management command may also be used to deallocate ranges of logical blocks by setting the Attribute – Deallocate (AD) field of Command Dword 11 to '1'. When the controller receives a Dataset Management command with the AD field set to '1', it may deallocate all provided ranges. If a read occurs to a deallocated range, the controller shall return all zeros, all ones, or the last data written to the associated LBA. If the deallocated or unwritten logical block error is enabled and a read occurs to a deallocated range, then the read shall fail with the Unwritten or Deallocated Logical Block status code.

An LBA that has been deallocated using the Dataset Management command is no longer deallocated when the LBA is written. Read operations do not affect the deallocation status of an LBA. The value read from a deallocated LBA shall be deterministic; specifically, the value returned by subsequent reads of that LBA shall be the same until a write occurs to that LBA.

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure or check for non-zero values in the DMRL, DMRSL, and DMSL fields to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue a Write command to write a known data pattern to the controller.
3. Configure the NVMe Host to issue a Dataset Management command with the Attribute – Deallocate (AD) field set to '1' and specifying the same LBA range written to in step 2 to the controller.
4. Configure the NVMe Host to issue a Read command to read the same LBA range written to in step 2.
5. Configure the NVMe Host to issue another Read command to the controller in order to read the same LBA range written to in step 2.
6. Configure the NVMe Host to issue a Write command to write a known data pattern (but different than the data pattern used in step 2) to the controller specifying the same LBA range previously deallocated.
7. Configure the NVMe Host to issue a Read command to the controller in order to read the same LBA range written to in step 6.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the data returned by the controller from the Read command in step 4 is either all zeros, all ones, or the data written to the associated LBA in step 2.
3. Verify that the data returned by the controller from the Read command in step 5 is identical to the data returned by the controller from the Read command in step 4.
4. Verify that the data returned by the controller from the Read command in step 7 matches the data written in step 6.

Case 3: **Deallocate Out of Range (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure or check for non-zero values in the DMRL, DMRSL, and DMSL fields to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue a Dataset Management command with the Attribute – Deallocate (AD) field set to '1' and specifying an out of Range LBA range value.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command, and the Dataset Management command completes with status LBA out of Range (80h).

Case 4: **NR Value is Maximum (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure or check for non-zero values in the DMRL, DMRSL, and DMSL fields to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue a Dataset Management command with the Number of Ranges (NR) field set to all '1's'.

Observable Results:

1. Verify that the command completes successfully.

Case 5: **Correct Range Deallocated (M, OF)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure or check for non-zero values in the DMRL, DMRSL, and DMSL fields to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Write a known data pattern (e.g. 'AAAA') to three consecutive LBAs.
3. Configure the NVMe Host to issue a Dataset Management command with the Attribute – Deallocate (AD) field set to '1' and specifying the middle LBA of the LBA written in step 1.
4. Perform a READ operation to each of the three LBAs.

Observable Results:

1. Verify that the Dataset Management command completes successfully.
2. Verify that the data returned for the READ command to the first and third LBAs is the known data pattern in step 1
3. Verify that the data returned from the READ operation to the second LBA is either all zeros, all ones, or the known data pattern from step 1.

Case 6: **Deallocate Multiple Ranges (M, OF)**

The Dataset Management command may also be used to deallocate ranges of logical blocks by setting the Attribute – Deallocate (AD) field of Command Dword 11 to '1'. When the controller receives a Dataset Management command with the AD field set to '1', it may deallocate all provided ranges. If a read occurs to a deallocated range, the controller shall return all zeros, all ones, or the last data written to the associated LBA. If the deallocated or unwritten logical block error is enabled and a read occurs to a deallocated range, then the read shall fail with the Unwritten or Deallocated Logical Block status code.

An LBA that has been deallocated using the Dataset Management command is no longer deallocated when the LBA is written. Read operations do not affect the deallocation status of an LBA. The value read from a deallocated LBA shall be deterministic; specifically, the value returned by subsequent reads of that LBA shall be the same until a write occurs to that LBA.

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure or check for non-zero values in the DMRL, DMRSL, and DMSL fields to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Configure the NVMe Host to issue a Write command to write a known data pattern to a range of LBAs in the controller.
3. Configure the NVMe Host to issue a Dataset Management command with the Attribute – Deallocate (AD) field set to '1' and specifying 3 separate smaller LBA ranges within the single larger LBA range written to in step 2 to the controller. The total capacity of the 3 smaller LBA ranges should be less than the capacity of the larger LBA range written in step 2, such that part of the larger range is not affected by the Dataset Management command.
4. Configure the NVMe Host to issue a Read command to read the same LBA range written to in step 2.

5. Configure the NVMe Host to issue another Read command to the controller in order to read the same LBA range written to in step 2.
6. Configure the NVMe Host to issue a Write command to write a known data pattern (but different than the data pattern used in step 2) to the controller specifying the same LBA range previously deallocated.
7. Configure the NVMe Host to issue a Read command to the controller in order to read the same LBA range written to in step 6.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the data returned by the controller from the Read command in step 4 is either all zeros, all ones, or the data written to the associated LBA in step 2.
3. Verify that the data returned by the controller from the Read command in step 5 is identical to the data returned by the controller from the Read command in step 4.
4. Verify that the data returned by the controller from the Read command in step 7 matches the data written in step 6.

Case 7: **NSID=FFFFFFFFh (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure or check for non-zero values in the DMRL, DMRSL, and DMSL fields to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Check the NVMe specification version supported by the DUT. If the specification is not v1.4 or higher, this test is not applicable.
3. Configure the NVMe Host to issue the Dataset Management command with NSID=FFFFFFFFh that indicates attributes for ranges of logical blocks.

Observable Results:

1. Verify that the Dataset Management Command does not complete successfully.

Case 8: **ONCS Bit 2 =1 Non-MDTS Command Size Limits DMRL (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports Dataset Management. If the DSM command is not supported this test is not applicable.
2. Check the DMRL values of the DUT.
3. Transmit a Data Set Management command in which NR (Number of Ranges) exceeds the DMRL value.

Observable Results:

1. Verify that the Dataset Management Command does not return status ‘Command Limit Exceeded’.

Case 9: **ONCS Bit 2 =1 DSM Supported Non-MDTS Command Size Limits DMRSL (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports Dataset Management. If the DSM command is not supported this test is not applicable.
2. Check the DMRL values of the DUT.
3. Transmit a Data Set Management command in which the number of logical blocks in. asingle range exceeds the DMRSL value.

Observable Results:

1. Verify that the Dataset Management Command does not return status ‘Command Limit Exceeded’.

Case 10: **ONCS Bit 2 =1 DSM Supported Non-MDTS Command Size Limits DMSL (FYI, OF-FYI)**

Test Procedure:
1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports Dataset Management. If the DSM command is not supported this test is not applicable.
2. Check the DMRL values of the DUT.
3. Transmit a Data Set Management command in which the total number of logical blocks in the command exceeds the DMSL value.

Observable Results:
1. Verify that the Dataset Management Command does not return status 'Command Limit Exceeded'.

Case 11: **ONCS Bit 2 =0 Non-MDTS Command Size Limits DMRL (FYI, OF-FYI)**

Test Procedure:
1. Check the ONCS Bit 2 of the Identify Controller Data structure. If ONCS Bit 2 is set to 1, this test is not applicable.
2. Check the DMRL values of the DUT.
3. Transmit a Data Set Management command in which NR (Number of Ranges) exceeds the DMRL value.

Observable Results:
1. Verify that the Dataset Management Command returns status 'Command Limit Exceeded'.

Case 12: **ONCS Bit 2 =0 DSM Supported Non-MDTS Command Size Limits DMRSL (FYI, OF-FYI)**

Test Procedure:
1. Check the ONCS Bit 2 of the Identify Controller Data structure. If ONCS Bit 2 is set to 1, this test is not applicable.
2. Check the DMRL values of the DUT.
3. Transmit a Data Set Management command in which the number of logical blocks in. asingle range exceeds the DMRSL value.

Observable Results:
1. Verify that the Dataset Management Command returns status 'Command Limit Exceeded'.

Case 13: **ONCS Bit 2 =0 DSM Supported Non-MDTS Command Size Limits DMSL (FYI, OF-FYI)**

Test Procedure:
1. Check the ONCS Bit 2 of the Identify Controller Data structure. If ONCS Bit 2 is set to 1, this test is not applicable.
2. Check the DMRL values of the DUT.
3. Transmit a Data Set Management command in which the total number of logical blocks in the command exceeds the DMSL value.

Observable Results:
1. Verify that the Dataset Management Command returns status 'Command Limit Exceeded'.

Possible Problems: None.

Test 2.3 – Read Command (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly execute the Read command.

References:

[1] NVMe Specification 6.9, 9.4, NVMe v1.3 ECN 001

Resource Requirements:

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

Last Modification: April 29, 2019

Discussion: The Read command reads data and metadata, if applicable, from the NVM controller for the LBAs indicated. The command may specify protection information to be checked as part of the read operation.

The command uses Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15 fields. If the command uses PRPs for the data transfer, then the Metadata Pointer, PRP Entry 1, and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the Metadata SGL Segment Pointer and SGL Entry 1 fields are used. If supported, this test may be performed using 4k sector sizes.

Regarding the reporting of error status codes, the NVMe specification states: "The status code of the completion queue entry should indicate an Internal Error status code (if multiple error conditions exist, the lowest numerical value is returned)."

Test Setup: See Appendix A.

Case 1: **Valid Read, LR=0, FUA=0 (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to one LBA on the NVMe Device.
2. Configure the NVMe Host to issue a Read command to the controller in order to read from the same LBA which was written to in step 1, with LR=0, FUA=0.
3. Verify that all received responses have all Reserved fields set to 0.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the known data pattern was read correctly from the NVMe Device exactly as it was written.

Case 2: **SLBA Out of Range (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
2. Configure the NVMe Host to issue a Read command to the controller to an SLBA which is out of range of the DUT.

Observable Results:

1. Verify that the READ command completes with status code LBA Out of Range (80h).

Case 3: **SLBA In Range, NLB Goes out of range (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
2. Configure the NVMe Host to issue a Read command to the controller to an SLBA which in range of the DUT, but an NLB value that will push the Read Command past the capacity of the DUT.

Observable Results:

1. Verify that the READ command completes with status code Invalid Field (02h) when the NLB value specifies a value that exceeds MDTS.
2. If NLB is out of range and does not exceed MDTS, verify that the READ command completes with status code LBA Out of Range (0x80).

Case 4: **SLBA Out of Range, NLB > MDTS (M, OF)**

Test Procedure:

1. Check the MDTS value reported by the DUT in the Identify Controller Data Structure. If MDTS is set to 0 (i.e. unlimited) then this test is not applicable.
2. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
3. Configure the NVMe Host to issue a Read command to the controller to an SLBA which is out of range of the DUT, and an NLB which is greater than MDTS.

Observable Results:

1. Verify that the READ command completes with status code of either Invalid Field (02h) or LBA out of Range (80h).

Case 5: **SLBA Out of Range, but Lower Dword = 00000000 (M. OF)**

Test Procedure:

1. If an SLBA of FFFFFFFF00000000h is within range of the DUT, then this test is not applicable.
2. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
3. Configure the NVMe Host to issue a Read command to the controller with an SLBA of FFFFFFFF00000000h ,which, when read as a 64 bit value is out of range of the DUT, and an NLB which is less than MDTS.

Observable Results:

1. Verify that the READ command completes with status code LBA Out of Range (80h).

Case 6: **Invalid Namespace ID (M, OF)**

Test Procedure:

1. Check the NN Value in the Identify Controller Data Structure. If NN is set to 0xFFFFFFFF, then this test Is not applicable.
2. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
3. Configure the NVMe Host to issue a Read command to the controller to the LBA 0000h.
4. Configure the NVMe Host to issue a READ command to the NVMe Controller specifying an Invalid NSID outside of the controller specified range defined by NN (Number of Namespaces).

Observable Results:

1. Verify that the READ command completes with status code Invalid Namespace or Format (0Bh).

Case 7: **Invalid Namespace ID and SLBA Out of Range (M, OF)**

Test Procedure:

1. Check the NN field in the Identify Controller Data Structure. If NN is the maximum possible value (0xFFFFFFFF) this sub test cannot be performed.

2. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
3. Configure the NVMe Host to issue a Read command to the controller to the LBA 0000h.
4. Configure the NVMe Host to issue a Read command to an invalid Namespace ID with an SLBA value that is beyond the capacity of the DUT. The READ command to the NVMe Controller specifying an Invalid NSID outside of the controller specified range defined by NN (Number of Namespaces).

Observable Results:

1. Verify that the READ command completes with status code of either Invalid Namespace or Format (0Bh), or LBA Out of Range (80h).

Case 8: **Valid Read, LR=0, FUA=1 (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to one LBA on the NVMe Device.
2. Configure the NVMe Host to issue a Read command to the controller in order to read from the same LBA which was written to in step 1, with LR=0, FUA=1.
3. Verify that all received responses have all Reserved fields set to 0.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the known data pattern was read correctly from the NVMe Device exactly as it was written.

Case 9: **Valid Read, LR=1, FUA=0 (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to one LBA on the NVMe Device.
2. Configure the NVMe Host to issue a Read command to the controller in order to read from the same LBA which was written to in step 1, with LR=1, FUA=0.
3. Verify that all received responses have all Reserved fields set to 0.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the known data pattern was read correctly from the NVMe Device exactly as it was written.

Case 10: **Valid Read, LR=1, FUA=1 (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to one LBA on the NVMe Device.
2. Configure the NVMe Host to issue a Read command to the controller in order to read from the same LBA which was written to in step 1, with LR=1, FUA=1.
3. Verify that all received responses have all Reserved fields set to 0.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the known data pattern was read correctly from the NVMe Device exactly as it was written.

Case 11: **NSID=FFFFFFFFh, LR=0, FUA=0 (FYI, OF-FYI)**

Test Procedure:

1. Check the NVMe specification version supported by the DUT. If the specification is not v1.4 or higher, this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to one LBA on the NVMe Device.
3. Configure the NVMe Host to issue a Read command to the controller with NSID=FFFFFFFFh in order to read from the same LBA which was written to in step 1, with LR=0, FUA=0.

Observable Results:

1. Verify that the READ command does not complete succesfully.

Possible Problems: None.

Test 2.4 – Write Command (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly execute the Write command.

References:

[1] NVMe Specification 6.14, 9.4

Resource Requirements:

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

Last Modification: November 14, 2017

Discussion: The Write command writes data and metadata, if applicable, to the NVM controller for the logical blocks indicated. The host may also specify protection information to include as part of the operation.

The command uses Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15 fields. If the command uses PRPs for the data transfer, then the Metadata Pointer, PRP Entry 1, and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the Metadata SGL Segment Pointer and SGL Entry 1 fields are used. If supported, this test may be performed using 4k sector sizes.

Regarding the reporting of error status codes, the NVMe specification states: "The status code of the completion queue entry should indicate an Internal Error status code (if multiple error conditions exist, the lowest numerical value is returned)."

Test Setup: See Appendix A.

Case 1: **Valid Write, LR=0, FUA=0 (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Write command to the controller in order to write a known data pattern to one LBA on the NVMe Device, with LR=0, FUA=0.
2. Configure the NVMe Host to issue a Read command to the controller in order to read from the same LBA which was written to in step 1.
3. Verify that all received responses have all Reserved fields set to 0.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the known data pattern was read correctly from the NVMe Device exactly as it was written.

Case 2: **SLBA Out of Range (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
2. Configure the NVMe Host to issue a Write command with a known data pattern to an SLBA which is out of range of the DUT.

Observable Results:

1. Verify that the Write command completes with status code LBA Out of Range (80h).

Case 3: **SLBA In Range, NLB Goes out of range (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
2. Configure the NVMe Host to issue a Write command with a known data pattern to an SLBA which in range of the DUT, but an NLB value that will push the Write Command past the capacity of the DUT.

Observable Results:

1. Verify that the WRITE command completes with status code Invalid Field (02h) when the NLB value specifies a value that exceeds MDTS.
2. If NLB is out of range and does not exceed MDTS, verify that the WRITE command completes with status code LBA Out of Range (0x80) or Capacity Exceeded (0x81).
3.

Case 4: **SLBA Out of Range, NLB > MDTS (M, OF)**

Test Procedure:

1. Check the MDTS value reported by the DUT in the Identify Controller Data Structure. If MDTS is set to 0 (i.e. unlimited) then this test is not applicable.
2. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
3. Configure the NVMe Host to issue a Write command with a known data pattern to an SLBA which is out of range of the DUT, and an NLB which is greater than MDTS.

Observable Results:

Verify that the Write command completes with status code of either Invalid Field (02h) or LBA out of Range (80h).

Case 5: **SLBA Out of Range, but Lower Dword = 00000000 (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
2. Configure the NVMe Host to issue a Write command with a known data pattern with an SLBA of FFFFFFFF00000000h ,which, when read as a 64 bit value is out of range of the DUT, and an NLB which is less than MDTS.

Observable Results:

1. Verify that the Write command completes with status code LBA Out of Range (80h).

Case 6: **Invalid Namespace ID (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.
2. Configure the NVMe Host to issue a Write command with a known data pattern, to an invalid Namespace ID. The Write command to the NVMe Controller should specify an Invalid NSID outside of the controller specified range defined by NN (Number of Namespaces). If NN is the maximum possible value (0xFFFFFFFF) this sub test cannot be performed.

Observable Results:

1. Verify that the Write command completes with status code Invalid Namespace or Format (0Bh).

Case 7: **Invalid Namespace ID and SLBA Out of Range (M, OF)**

Test Procedure:

1. Check the NN field in the Identify Controller Data Structure. If NN is the maximum possible value (0xFFFFFFFF) this sub test cannot be performed.
2. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.

3. Configure the NVMe Host to issue a Write command with a known data pattern, to an invalid Namespace ID with an SLBA value that is beyond the capacity of the DUT. The Write command to the NVMe Controller should specify an Invalid NSID outside of the controller specified range defined by NN (Number of Namespaces).

Observable Results:

1. Verify that the Write command completes with status code of either Invalid Namespace or Format (0Bh), or LBA Out of Range (80h).

Case 8: **Valid Write, LR=0, FUA=1 (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Write command with a known data pattern, to the controller in order to write a known data pattern to one LBA on the NVMe Device, with LR=0, FUA=1.
2. Configure the NVMe Host to issue a Read command to the controller in order to read from the same LBA which was written to in step 1.
3. Verify that all received responses have all Reserved fields set to 0.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the known data pattern was read correctly from the NVMe Device exactly as it was written.

Case 9: **Valid Write, LR=1, FUA=0 (M)**

Test Procedure:

1. Configure the NVMe Host to issue a Write command with a known data pattern, to the controller in order to write a known data pattern to one LBA on the NVMe Device, with LR=1, FUA=0.
2. Configure the NVMe Host to issue a Read command to the controller in order to read from the same LBA which was written to in step 1.
3. Verify that all received responses have all Reserved fields set to 0.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the known data pattern was read correctly from the NVMe Device exactly as it was written.

Case 10: **Valid Write, LR=1, FUA=1 (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Write command with a known data pattern, to the controller in order to write a known data pattern to one LBA on the NVMe Device, with LR=1, FUA=1.
2. Configure the NVMe Host to issue a Read command to the controller in order to read from the same LBA which was written to in step 1.
3. Verify that all received responses have all Reserved fields set to 0.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the known data pattern was read correctly from the NVMe Device exactly as it was written.

Case 11: **NSID=FFFFFFFFh, LR=0, FUA=0 (FYI, OF-FYI)**

Test Procedure:

1. Check the NVMe specification version supported by the DUT. If the specification is not v1.4 or higher, this test is not applicable.

2. Configure the NVMe Host to issue a Write command to the controller with NSID=FFFFFFFFh in order to write a known data pattern to one LBA on the NVMe Device, with LR=0, FUA=0.
3. Configure the NVMe Host to issue a Read command to the controller in order to read from the same LBA which was written to in step 1.

Observable Results:

1. Verify that the Write command did not complete successfully and the known data pattern was not read back in the subsequent READ operation.

Possible Problems: None.

Test 2.5 – Write Uncorrectable Command (M, OF)

Purpose: To verify that an NVMe Controller can properly execute the Write Uncorrectable command.

References:

[1] NVMe Specification 6.15

Resource Requirements:

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

Last Modification: August 29, 2017

Discussion: The Write Uncorrectable command is used to mark a range of logical blocks as invalid. When the specified logical blocks(s) are read after this operation, a failure is returned with Unrecovered Read Error status. To clear the invalid logical block status, a write operation is performed for those logical blocks.

The fields used are Command Dword 10, Command Dword 11, and Command Dword 12 fields. All other command specific fields are reserved.

Test Setup: See Appendix A.

Case 1: **SLBA In Range, NLB Valid (M, OF)**

Test Procedure:

1. Check the ONCS field or check for a non-zero value in the WUSL field to determine if the DUT supports the Write Uncorrectable command. If the DUT does not support the Write Uncorrectable command this test is not applicable.
2. Configure the NVMe Host to issue a Write Uncorrectable command to a particular LBA on the NVMe Device. Both the SLBA and NLB should be valid.
3. Configure the NVMe Host to issue a Read command for the LBA on which the Write Uncorrectable command was performed.
4. Configure the NVMe Host to issue a Write command for the LBA on which the Write Uncorrectable command was performed in order to clear the invalid logical block status.
5. Repeat step 3.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the completion queue entry returned for the first Read command indicates *Unrecovered Read Error* status.
3. Verify that the completion queue entry returned for the second Read command indicates Success status.

Case 2: **SLBA Out of Range, NLB Valid (M, OF)**

Test Procedure:

1. Check the ONCS field or check for a non-zero value in the WUSL field to determine if the DUT supports the Write Uncorrectable command. If the DUT does not support the Write Uncorrectable command this test is not applicable.
2. Configure the NVMe Host to issue a Write Uncorrectable command to a particular LBA on the NVMe Device. The SLBA should be for a value out of range for the DUT. NLB should be valid.
3. Configure the NVMe Host to issue a Read command for the LBA on which the Write Uncorrectable command was performed.

Observable Results:

1. Verify that that the Write Uncorrectable Command returns status code LBA Out of Range 80h.

2. Verify that the completion queue entry returned for the first Read command indicates Status Code LBA out of Range and does not return status code Unrecovered READ error.

Case 3: **SLBA Out of Range, NSID Invalid (M, OF)**

Test Procedure:

1. Check the ONCS field or check for a non-zero value in the WUSL field to determine if the DUT supports the Write Uncorrectable command. If the DUT does not support the Write Uncorrectable command this test is not applicable.
2. Check the NN value reported by the DUT in the Identify Controller Data Structure. If NN is the maximum possible value (0xFFFFFFFF) this sub test cannot be performed.
3. Configure the NVMe Host to issue a Write Uncorrectable command to a particular LBA on the NVMe Device. The SLBA should be for a value out of range for the DUT, and the NSID should be invalid. The Write Uncorrectable command to the NVMe Controller should specify an Invalid NSID outside of the controller specified range defined by NN (Number of Namespaces).
4. Configure the NVMe Host to issue a Read command for the LBA on which the Write Uncorrectable command was performed.

Observable Results:

1. Verify that the Write Uncorrectable command completes with status code Invalid Namespace or Format (0Bh), and not LBA Out of Range (80h).
2. Verify that the completion queue entry returned for the first Read command indicates Status Code Invalid Namespace or Format and does not return status code Unrecovered READ error.

Case 4: **SLBA Out of Range, but Lower Dword = 00000000 (M, OF)**

Test Procedure:

1. Check the ONCS field or check for a non-zero value in the WUSL field to determine if the DUT supports the Write Uncorrectable command. If the DUT does not support the Write Uncorrectable command this test is not applicable.
2. Configure the NVMe Host to issue a Write Uncorrectable command to a particular LBA on the NVMe Device. The SLBA should be FFFFFFFF00000000h ,which, when read as a 64 bit value is out of range of the DUT, and an NLB which is less than MDTS.
3. Configure the NVMe Host to issue a Read command for the LBA on which the Write Uncorrectable command was performed.

Observable Results:

1. Verify that the Write Uncorrectable command completes with status code LBA Out of Range (80h).
2. Verify that the completion queue entry returned for the first Read command indicates Status Code LBA out of Range and does not return status code Unrecovered READ error.

Case 5: **NLB greater than MDTS and Non-MDTS Command Size Limits Not Supported(M, OF)**

Test Procedure:

1. Check the DMRL, DMRSL, and DMSL fields. If these fields are set to non zero values this test is not applicable.
2. Check the ONCS field to determine if the DUT supports the Write Uncorrectable command. If the DUT does not support the Write Uncorrectable command this test is not applicable.
3. Check the MDTS value reported by the DUT in the Identify Controller Data Structure. If MDTS is set to 0 (unlimited) then this test case is not applicable.
4. Configure the NVMe Host to issue a Write Uncorrectable command to a particular LBA on the NVMe Device. NLB should be greater than MDTS.
5. Configure the NVMe Host to issue Read commands for the LBAs on which the Write Uncorrectable command was performed. Ensure that the NVMe Host issues Read commands to all LBAs affected by the Write Uncorrectable command.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status of Successful for the Write Uncorrectable command. The DUT is expected to ignore the NLB/MDTS conflict since MDTS does not affect the Write Uncorrectable command.
2. Verify that the completion queue entry returned for the Read commands indicate *Unrecovered Read Error* status.

Case 6: **NSID=FFFFFFFFh, SLBA In Range, NLB Valid (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field or check for a non-zero value in the WUSL field to determine if the DUT supports the Write Uncorrectable command. If the DUT does not support the Write Uncorrectable command this test is not applicable.
2. Check the NVMe specification version supported by the DUT. If the specification is not v1.4 or higher, this test is not applicable.
3. Configure the NVMe Host to issue a Write Uncorrectable command with NSID=FFFFFFFFh to a particular LBA on the NVMe Device. Both the SLBA and NLB should be valid.
4. Configure the NVMe Host to issue a Read command for the LBA on which the Write Uncorrectable command was performed.
5. Configure the NVMe Host to issue a Write command for the LBA on which the Write Uncorrectable command was performed in order to clear the invalid logical block status.
6. Repeat step 3.

Observable Results:

1. Verify that the Write Uncorrectable command does not complete successfully.
2. Verify that the completion queue entry returned for both Read commands indicate Success status.

Case 7: **NLB greater than WUSL and Non-MDTS Command Size Limits Supported(FYI, OF-FYI)**

Test Procedure:

1. Check the DMRL, DMRSL, and DMSL fields. If these fields are set to 0h this test is not applicable.
2. Check the ONCS field to determine if the DUT supports the Write Uncorrectable command. If the DUT does not support the Write Uncorrectable command this test is not applicable.
3. Check the WUSL value reported by the DUT in the Identify Controller Data Structure. If WUSL is set to 0 (unlimited) then this test case is not applicable.
4. Configure the NVMe Host to issue a Write Uncorrectable command to a particular LBA on the NVMe Device. NLB should be greater than WUSL.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status of Invalid Field in command. for the Write Uncorrectable command.

Possible Problems:

Case 5 is updated per ECN 003 of NVMe Specification v1.2.1. This ECN alters behavior expected in earlier versions.

Test 2.6 – Flush Command (M, OF)

Purpose: To verify that an NVMe Controller can properly execute the Flush command.

References:

[1] NVMe Specification 6.8, TP 4035

Resource Requirements:

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

Last Modification: April 24, 2019

Discussion: The Flush command shall commit data and metadata associated with the specified namespace(s) to non–volatile media. The flush applies to all commands completed prior to the submission of the Flush command. The controller may also flush additional data and/or metadata from any namespace.

All command specific fields are reserved.

Test Setup: See Appendix A.

Case 1: **Valid Namespace ID (M, OF)**

Test Procedure:

1. For each active namespace, configure the NVMe Host to issue a Flush command to the NVMe Controller specifying the NSID of the specified namespace.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. If the VWC field Bit 0 of the Identify Controller data structure is cleared to 0, verify that the completion queue entry for each Flush command indicate success.
3. Verify that all received responses have all Reserved fields set to 0.

Case 2: **Invalid Namespace ID (M, OF)**

Test Procedure:

1. Configure the NVMe Host to issue a Flush command to the NVMe Controller specifying an Invalid NSID outside of the controller specified range defined by NN (Number of Namespaces). If NN is the maximum possible value (0xFFFFFFFF) this sub test cannot be performed.

Observable Results:

1. Verify that the Flush command completes with status code Invalid Namespace or Format (0Bh).

Case 3: **NSID=0xFFFFFFFF (M, OF)**

Test Procedure:

1. Check the VWC field of the Identify Controller Data Structure.
2. Configure the NVMe Host to issue a Flush command to the NVMe Controller specifying an NSID of 0xFFFFFFFF.

Observable Results:

1. If VWC bits 2:1 is 00b, verify that the DUT indicates support for version 1.3 or earlier of the NVMe specification in the VER field.
2. If VWC bits 2:1 is 10b, verify that controller fails the command with status code Invalid Namespace or Format.

3. If VWC bits 2:1 is 11b, verify that the Flush command completes with status success, and is applied to all namespaces attached to the controller.
4. Verify that VWC bits 2:1 is not set to 01b.

Possible Problems: None.

Test 2.7 – Write Zeroes Command (M, OF)

Purpose: To verify that an NVMe Controller can properly execute the Write Zeroes command.

References:

[1] NVMe Specification 6.16

Resource Requirements:

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

Last Modification: November 20, 2018

Discussion: The Write Zeroes command is used to set a range of logical blocks to zero. After successful completion of this command, the value returned by subsequent reads of logical blocks in this range shall be zeroes until a write occurs to this LBA range. The metadata for this command shall be all zeroes and the protection information is updated based on CDW12.PRINFO.

The fields used are Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 14, and Command Dword 15 fields.

Test Setup: See Appendix A.

Case 1: **SLBA In Range, NLB Valid, LR=0, FUA=0 (M, OF)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data Structure or check for a non-zero value in the WZSL field to determine if the Write Zeroes command is supported. If the Write Zeroes Command is not supported then this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller specifying a particular LBA on the NVMe Device. The written data should be either all 1's, or an alternating pattern of 1's and 0's.
3. Configure the NVMe Host to issue a Write Zeroes command to the controller specifying the same LBA on the NVMe Device which was previously written to.
4. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA for which the Write Zeroes command was performed.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the data returned by the controller for the Read command after the Write Zeroes command returns data of zeroes only.

Case 2: **SLBA Out of Range, NLB Valid (M, OF)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data Structure or check for a non-zero value in the WZSL field to determine if the Write Zeroes command is supported. If the Write Zeroes Command is not supported then this test is not applicable.
2. Configure the NVMe Host to issue a Write Zeroes command to the controller specifying SLBA which is out of range for the DUT.

Observable Results:

1. Verify that that the Write Zeroes Command returns status code LBA Out of Range 80h.

Case 3: **SLBA Out of Range, NSID Invalid (M, OF)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data Structure or check for a non-zero value in the WZSL field to determine if the Write Zeroes command is supported. If the Write Zeroes Command is not supported then this test is not applicable.
2. Check the NN value reported by the DUT in the Identify Controller Data Structure. If NN is the maximum possible value (0xFFFFFFFF) this sub test cannot be performed.
3. Configure the NVMe Host to issue a Write Zeroes command to the controller specifying SLBA which is out of range for the DUT, and an invalid NSID. The Write Zeroes command to the NVMe Controller should specify an Invalid NSID outside of the controller specified range defined by NN (Number of Namespaces).

Observable Results:

1. Verify that the Write Zeroes command completes with status code or either Invalid Namespace or Format (0Bh or LBA Out of Range (80h).

Case 4: **SLBA Out of Range, but Lower Dword = 00000000 (M, OF)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data Structure or check for a non-zero value in the WZSL field to determine if the Write Zeroes command is supported. If the Write Zeroes Command is not supported then this test is not applicable.
2. Configure the NVMe Host to issue a Write Zeroes command to a particular LBA on the NVMe Device. The SLBA should be FFFFFFFF00000000h ,which, when read as a 64 bit value is out of range of the DUT, and an NLB which is less than MDTS.

Observable Results:

1. Verify that the Write Zeroes command completes with status code LBA Out of Range (80h).

Case 5: **NLB greater than MDTS and Non-MDTS Command Size Limits Not Supported (M, OF)**

Test Procedure:

1. Check the DMRL, DMRSL, and DMSL fields. If these fields are set to non zero values this test is not applicable.
2. Check the ONCS field of the Identify Controller Data Structure. If the Write Zeroes Command is not support then this test is not applicable.
3. Check the MDTS value reported in the Identify Controller Data Structure, If MDTS is set to 0 (unlimited) then this test case is not applicable.
4. Configure the NVMe Host to issue a Write command to the controller specifying a particular LBA on the NVMe Device. The written data should be either all 1's, or an alternating pattern of 1's and 0's.
5. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA for which the previous Write command was performed.
6. Configure the NVMe Host to issue a Write Zeroes command the same LBA on the NVMe Device as was written in Step 1. NLB should be greater than MDTS.
7. Configure the NVMe Host to issue Read commands to the controller specifying the same LBAs for which the Write Zeroes command was performed. Ensure that the NVMe Host issues Read commands to all LBAs affected by the Write Zeroes command.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command which should be 'Succesful'.
2. Verify that the data returned by the controller for the Read command after the Write command returns the data written in Step 1.
3. Verify that the data returned by the controller for the Read commands after the Write Zeroes command returns all 0's.

Case 6: **SLBA In Range, NLB Valid, LR=0, FUA=1 (M, OF)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data Structure or check for a non-zero value in the WZSL field to determine if the Write Zeroes command is supported. If the Write Zeroes Command is not supported then this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller specifying a particular LBA on the NVMe Device. The written data should be either all 1's, or an alternating pattern of 1's and 0's.
3. Configure the NVMe Host to issue a Write Zeroes command to the controller specifying the same LBA on the NVMe Device which was previously written to.
4. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA for which the Write Zeroes command was performed.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the data returned by the controller for the Read command after the Write Zeroes command returns data of zeroes only.

Case 7: **SLBA In Range, NLB Valid, LR=1, FUA=0 (M, OF)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data Structure or check for a non-zero value in the WZSL field to determine if the Write Zeroes command is supported. If the Write Zeroes Command is not supported then this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller specifying a particular LBA on the NVMe Device. The written data should be either all 1's, or an alternating pattern of 1's and 0's.
3. Configure the NVMe Host to issue a Write Zeroes command to the controller specifying the same LBA on the NVMe Device which was previously written to.
4. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA for which the Write Zeroes command was performed.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the data returned by the controller for the Read command after the Write Zeroes command returns data of zeroes only.

Case 8: **SLBA In Range, NLB Valid, LR=1, FUA=1 (M, OF)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data Structure or check for a non-zero value in the WZSL field to determine if the Write Zeroes command is supported. If the Write Zeroes Command is not supported then this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller specifying a particular LBA on the NVMe Device. The written data should be either all 1's, or an alternating pattern of 1's and 0's.
3. Configure the NVMe Host to issue a Write Zeroes command to the controller specifying the same LBA on the NVMe Device which was previously written to.
4. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA for which the Write Zeroes command was performed.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that the data returned by the controller for the Read command after the Write Zeroes command returns data of zeroes only.

Case 9: **PRCHK is Non Zero (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data Structure or check for a non-zero value in the WZSL field to determine if the Write Zeroes command is supported. If the Write Zeroes Command is not supported then this test is not applicable.
2. Check for support for End to End Data Protection by checking the DPC field in the Identify Namespace Data Structure. If the DUT does not support End to End Data Protection, this test is not applicable.
3. Perform a Format NVM command to enable the namespace to use end to end data protection. End to End data protection must align with the support indicated in the DPC field.
4. Configure the NVMe Host to issue a Write command to the controller specifying the namespace with end to end data protection on the NVMe Device. The written data should be either all 1's, or an alternating pattern of 1's and 0's.
5. Configure the NVMe Host to issue a Write Zeroes command to the controller specifying the same LBA on the NVMe Device which was previously written to, but the PRCHK value within the PRINFO field is non-zero.
6. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA for which the Write Zeroes command was performed.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command, and that the Write Zeroes command completes with Status 02h Invalid Field or Status 81h Invalid Protection Information.
2. Verify that the data returned by the controller for the Read command after the Write Zeroes command returns the data pattern written in step 1.

Case 10: **NSID=FFFFFFFFh, SLBA In Range, NLB Valid, LR=0, FUA=0 (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data Structure or check for a non-zero value in the WZSL field to determine if the Write Zeroes command is supported. If the Write Zeroes Command is not supported then this test is not applicable.
2. Check the NVMe specification version supported by the DUT. If the specification is not v1.4 or higher, this test is not applicable.
3. Configure the NVMe Host to issue a Write command to the controller specifying a particular LBA on the NVMe Device. The written data should be either all 1's, or an alternating pattern of 1's and 0's.
4. Configure the NVMe Host to issue a Write Zeroes command with NSID=FFFFFFFFh to the controller specifying the same LBA on the NVMe Device which was previously written to.
5. Configure the NVMe Host to issue a Read command to the controller specifying the same LBA for which the Write Zeroes command was performed.

Observable Results:

1. Verify that the Write Zeroes command did not complete succesfully.
2. Verify that the data returned by the controller for the Read command after the Write Zeroes command returns the same data as was previously written.

Case 11: **NLB greater than WZSL and Non-MDTS Command Size Limits Supported (FYI, OF-FYI)**

Test Procedure:

1. Check the DMRL, DMRSL, and DMSL fields. If these fields are set to 0h this test is not applicable.
2. Check the ONCS field of the Identify Controller Data Structure. If the Write Zeroes Command is not supported then this test is not applicable.
3. Check the WZSL value reported in the Identify Controller Data Structure, If WZSL is set to 0 (unlimited) this test case is not applicable.
4. Configure the NVMe Host to issue a Write Zeroes command with NLB greater than WZSL.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command of 'Invalid Field in Command'.

Possible Problems: Case 5 is updated per ECN 003 of NVMe Specification v1.2.1. This ECN alters behavior expected in earlier versions.

Test 2.8 – Atomicity Parameters (M, OF)

Purpose: To verify that an NVMe Controller properly sets the Atomicity Parameters.

References:

[1] NVMe Specification 6.4

Resource Requirements:

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

Last Modification: November 28, 2017

Discussion: The NVMe specification sets certain restrictions on the Atomicity Parameters relative to one another, as reported in the Identify Controller and Identify Namespace data structures. This tests seeks to ensure that those restrictions are followed. These restrictions apply to the AWUN, AWUPF, ACWU, NAWUN, NAWUPF, NACWU, NABSN, NABO, NABSPF values. Support for Atomic Writes is indicated in the NSFEAT and NAWUN fields. If the DUT does not support Atomic Writes this test is not applicable and does not need to be performed.

Test Setup: See Appendix A.

Case 1: **NVM Command Set Supported (M, OF)**

Test Procedure:

1. For each namespace in the NVM subsystem, configure the NVMe Host to issue an Identify command specifying CNS value 00h to the controller in order to receive back an Identify Namespace data structure for the specified namespace.
2. Check that Bit 37 of CAP.CSS is set to 1, otherwise this test is not applicable.
3. Configure the NVMe Host to issue an Identify command specifying CNS value 01h to the controller in order to receive back an Identify Controller data structure.
4. Parse the received data structures for the AWUN, AWUPF, ACWU, NAWUN, NAWUPF, NACWU, NABSN, NABO, NABSPF values.

Observable Results:

1. Determine if NABSN is set to a non-zero value. If not, this test case is not applicable.
2. Verify the following:
    a. AWUPF <= AWUN
    b. NAWUN >= AWUN
    c. NAWUPF >= AWUPF
    d. NAWUPF<= NAWUN
    e. NACWU>=ACWU
    f. NABSN>=NAWUN
    g. NABO<=NABSN
    h. NABO<=NABSPF
    i. NABSPF>=NAWUPF

Case 2: **NVM Command Set Not Supported (FYI, OF-FYI)**

Test Procedure:

1. Check that Bit 37 of CAP.CSS is set to 0, otherwise this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 01h to the controller in order to receive back an Identify Controller data structure.
3. Parse the received data structures for the AWUN, AWUPF, ACWU.

Observable Results:

1. Verify that AWUN, AWUPF, and ACWU are set to 0.

Possible Problems: None.

Test 2.9 – AWUN/NAWUN (M)

Purpose: To verify that an NVMe Controller properly uses its Atomicity Parameters.

References:

[1] NVMe Specification 6.4.1

Resource Requirements:

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

Last Modification: August 29, 2017

Discussion: AWUN/NAWUN control the atomicity of command execution in relation to other commands. They impose inter-command serialization of writing of blocks of data to the NVM and prevent blocks of data ending up on the NVM containing partial data from one new command and partial data from one or more other new commands.

If a write command is submitted with size less than or equal to the AWUN/NAWUN value and the write command does not cross an atomic boundary, then the host is guaranteed that the write command is atomic to the NVM with respect to other read or write commands. If a write command is submitted with size greater than the AWUN/NAWUN value or crosses an atomic boundary, then there is no guarantee of command atomicity. AWUN/NAWUN does not have any applicability to write errors caused by power failure or other error conditions.

Test Setup: See Appendix A.

Case 1: **Atomic Boundaries Not Supported (NABSN/NABSPF = 0) (M)**

Test Procedure:

1. Ensure that both NABSN and NABSPF are set to 0. If not, this test case is not applicable and Case 2 below should be performed.
2. Configure the NVMe Host to issue 2 Write Commands, each with a length equal to 4 logical blocks. (i.e. if the logical block size is 512 bytes, the Write would be 2K in length. If the logical block size is 4K, the length of the Write would be 16K). The first command (Command A) will write Logical Blocks 0-3, with an FFFFh pattern. The second command (Command B) will write to Logical Blocks 1-4, with an AAAAh pattern.
3. Repeat for all supported Namespaces.

Observable Results:

1. Verify that one of the following has occurred, any other outcome is a failure:
    a. LBA 0-3 have an FFFFh pattern and LBA 4 has an AAAAh pattern.
    b. LBA 0 has an FFFFh pattern and LBA 1-4 has an AAAAh pattern.

Case 2: **Atomic Boundaries Supported (NABSN ≠ 0) (M)**

Test Procedure:

1. Determine if NABSN is set to a non-zero value. If not, this test case is not applicable and Case 1 above should be performed.
2. Configure the NVMe Host to issue 2 Write Commands, each with a length of half of NABSN. The first command (Command A) will write 4x sets of length NABSN, with an FFFFh pattern. The second command (Command B) will write 4x sets of length NABSN with an AAAAh pattern. Command B will be offset from Command A by half of NABSN. In total 5 sets of data with length ½ NABSN will be written.
3. Repeat for all supported Namespaces.

Observable Results:

1. Verify that one of the following has occurred, any other outcome is a failure:
    a. The first 4x NABSN of data is a FFFFh pattern and the last 1x NABSN of data is an AAAAh pattern.

b. The first 1x NABSN of data is a FFFFh pattern and the last 4x NABSN of data is an AAAAh pattern.

Possible Problems: For Case 2 above, NABSPF could be set to a non-zero value indicating support for Atomic Boundaries during a Power Failure. However, test capability has not been developed for testing Atomic Writes during Power Failure, so only the Normal condition is tested.

Test 2.10 – AWUPF/NAWUPF (IP)

Purpose: To verify that an NVMe Controller properly uses its Atomicity Parameters when a power failure occurs.

References:

[1] NVMe Specification 6.4.2

Resource Requirements:

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

Last Modification: June 13, 2016

Discussion: AWUPF and NAWUPF indicate the behavior of the controller if a power fail or other error condition interrupts a write operation causing a torn write. A torn write is a write operation where only some of the logical blocks that are supposed to be written contiguously are actually stored on the NVM, leaving the target logical blocks in an indeterminate state in which some logical blocks contain original data and some logical blocks contain new data from the write operation.

If a write command is submitted with size less than or equal to the AWUPF/NAWUPF value and the write command does not cross an atomic boundary the controller guarantees that if the command fails due to a power failure or other error condition, then subsequent read commands for the logical blocks associated with the write command shall return one of the following:

- All old data (i.e. original data on the NVM in the LBA range addressed by the interrupted write), or
- All new data (i.e. all data to be written to the NVM by the interrupted write)

If a write command is submitted with size greater than the AWUPF/NAWUPF value or crosses an atomic boundary, then there is no guarantee of the data returned on subsequent reads of the associated logical blocks.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the DUT to use AWUPF/NAWUPF values to be greater than the size of 2 logical blocks.
2. Perform a READ operation to Logical Blocks 0-1.
3. Configure the NVMe Host to issue a Write Command, with a length equal to 2 logical blocks. (i.e. if the logical block size is 512 bytes, the Write would be 1K in length. If the logical block size is 4K, the length of the Write would be 8K). The command (Command A) will write Logical Blocks 0-1, with an AAAAh pattern. Before command completion, cause a Power Failure event.
4. Allow the PCIe link to reset, and for the NVMe Controller to return to the enabled state.
5. Perform a READ operation to Logical Blocks 0-1.

Observable Results:

1. Verify that one of the following has occurred, any other outcome is a failure:
    a. Contents of LBA 0-1 have an AAAAh pattern.
    b. Contents of LBA 0-1 are unchanged.

Possible Problems: Tools for reliably creating a power failure during a WRITE operation are not available. This test will remain an FYI test until such tools are available.

Test 2.11 – Verify Command (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly execute the Verify command, if supported.

References:

[1] NVMe Specification TP 4030

Resource Requirements:

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

Last Modification: April 24, 2019

Discussion: The Verify command is roughly equivalent to a Read command that discards the data (and metadata) after reading. The important behavior of Verify is that errors are reported if the data (or metadata) cannot be read without expending the time and resources required to return the data (and metadata) to the host.

Test Setup: See Appendix A.

Case 1: **Valid Command (FYI, OF-FYI)**

Test Procedure:

1. Check ONCS bit 7 or for a non-zero value in the VSL field to determine if the DUT supports the Verify Command. If the Verify Command is not supported, this test is not applicable.
2. Perform a Verify command of LBADS bytes. If the namespace is formatted with protection information, perform the Verify Command using the same protection information as the namespace was formatted with, and PRACT=0.
3. Repeat for all supported PRINFO values. It may be necessary to format the namespace appropriately to check all Protection Information types.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. Verify that all received responses have all Reserved fields set to 0.

Case 2: **PRACT = 1 (FYI, OF-FYI)**

Test Procedure:

1. Check ONCS bit 7 or for a non-zero value in the VSL field to determine if the DUT supports the Verify Command. If the Verify Command is not supported, this test is not applicable.
2. Perform a Verify command, using the same protection information as the namespace was formatted with, and with PRACT = 1.

Observable Results:

1. Verify that the Verify command completes with status Invalid Field in Command.

Case 3: **NSID=FFFFFFFFh (M, OF-FYI)**

Test Procedure:

1. Check ONCS bit 7 or for a non-zero value in the VSL field to determine if the DUT supports the Verify Command. If the Verify Command is not supported, this test is not applicable.
2. Check the NVMe specification version supported by the DUT. If the specification is not v1.4 or higher, this test is not applicable.
3. Perform a Verify command of LBADS bytes and NSID=FFFFFFFFh. If the namespace is formatted with protection information, perform the Verify Command using the same protection information as the namespace was formatted with, and PRACT=0.
4. Repeat for all supported PRINFO values. It may be necessary to format the namespace appropriately to check all Protection Information types.

Observable Results:

1. Verify that none of the Verify commands complete successfully.

Case 4: **Command Size Limits (FYI, OF-FYI)**

Test Procedure:

1. Check ONCS bit 7, if this bit is set to 1 this test is not applicable.
2. Check the VSL field in the Identify Controller Data structure. If VSL field is set to 0 this test is not applicable.
3. Perform a Verify command with an NLB value that exceeds VSL.

Observable Results:

1. Verify that the Verify command completes with status 'Invalid Field in Command'.

Possible Problems: None.

Test 2.12 – Fused Operations (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller can properly execute Fused Operations.

References:

[1] NVMe Specification 6.2

Resource Requirements:

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

Last Modification: November 13, 2019

Discussion: The Compare and Write fused operation compares the contents of the logical block(s) specified in the Compare command to the data stored at the indicated LBA range. If the compare is successful, then the LBA range is updated with the data provided in the Write command. If the Compare operation is not successful, then the Write operation is aborted with a status of Command Aborted due to Failed Fused Command and the contents in the LBA range are not modified. If the Write operation is not successful, the Compare operation completion status is unaffected.

Test Setup: See Appendix A.

Case 1: **Supported Fused Operation (FYI, OF-FYI)**

Test Procedure:

1. Check the FUSES field of the Identify Controller Data structure to determine if the DUT supports Fused Operations. If fused operations are not supported this test is not applicable.
2. Configure the NVMe Host to issue a Compare command with the FUSE field set to 01b, to indicate the first command in a fused operation followed by a Write command with the FUSE field set to 10b to indicate the second command in a fused operation to the controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the Compare and Write commands complete successfully.

Case 2: **Fused Operations not supported (FYI, OF-FYI)**

Test Procedure:

1. Check the FUSES field of the Identify Controller Data structure to determine if the DUT supports Fused Operations. If fused operations are supported this test is not applicable.
2. Configure the NVMe Host to issue a Compare command with the FUSE field set to 01b, to indicate the first command in a fused operation followed by a Write command with the FUSE field set to 10b to indicate the second command in a fused operation to the controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that both the Compare and Write commands complete with status ‘Invalid Field in Command’.

Case 3: **Missing Fused Command (FYI, OF-FYI)**

Test Procedure:

1. Check the FUSES field of the Identify Controller Data structure to determine if the DUT supports Fused Operations. If fused operations are not supported this test is not applicable.
2. Verify that the DUT claims support for NVME v1.4 or higher. If not, then this test is not applicable.
3. Configure the NVMe Host to issue a Compare command with the FUSE field set to 01b, to indicate the first command in a fused operation. Wait for the command to complete.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the Compare command was aborted with status 'Command Aborted due to Missing Fused Command'.

Case 4: **LBA Range Mismatch (FYI, OF-FYI)**

Test Procedure:

1. Check the FUSES field of the Identify Controller Data structure to determine if the DUT supports Fused Operations. If fused operations are not supported this test is not applicable.
2. Configure the NVMe Host to issue a Compare command with the FUSE field set to 01b, to indicate the first command in a fused operation followed by a Write command with the FUSE field set to 10b to indicate the second command in a fused operation to the controller. The Write command should indicate a different LBA range than the Compare Command.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that both the Compare and Write commands complete with status 'Invalid Field in Command'.

Case 5: **Unsupported Fused Operation (FYI, OF-FYI)**

Test Procedure:

1. Check the FUSES field of the Identify Controller Data structure to determine if the DUT supports Fused Operations. If fused operations are not supported this test is not applicable.
2. Configure the NVMe Host to issue a Read command with the FUSE field set to 01b, to indicate the first command in a fused operation.

Observable Results:

1. Verify that the Read command is aborted with status 'Invalid Field in Command'.

Possible Problems: None.

Test 2.13 – Deallocate and Allocate (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller can properly operate depending on the DULBE setting.

References:

[1] NVMe Specification 6.7

Resource Requirements:

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

Last Modification: May 11, 2020

Discussion: A logical block that has never been written to, or which has been deallocated using the Dataset Management command, the Write Zeroes command or the Sanitize command is called a deallocated or unwritten logical block.

.

Test Setup: See Appendix A.

Case 1: **Deallocate via Dataset Management, DULBE=0 (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Perform an Identify Namespace Data Structure Command and record the value of the DLFEAT field.
3. Perform a Set Feature command for FID 05h, Error Recovery Feature, to set DULBE=0.
4. Perform a Get Feature command for FID 05h, Error Recovery Feature, to ensure that DULBE=0.
5. Configure the NVMe Host to issue a Write command to write a known data pattern to the controller.
6. Configure the NVMe Host to issue a Dataset Management command with the Attribute – Deallocate (AD) field set to '1' and specifying the same LBA range written to in the previous step.
7. Configure the NVMe Host to issue 2 Read commands to the same LBA range deallocated in the previous step.

Observable Results:

1. Verify that the same data pattern is returned to each Read command.
2. Verify that the Read data returned has all bytes cleared to 0h if bits 2:0 in the DLFEAT field are set to 001b.
3. Verify that the Read data returned has all bytes set to FFh if bits 2:0 in the DLFEAT field are set to 010b.
4. Verify that the Read data returned has either all bytes cleared to 0h or all bytes set to FFh if bits 2:0 in the DLFEAT field are set to 000b.

Case 2: **Deallocate via Dataset Management, DULBE=1 (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Perform a Set Feature command for FID 05h, Error Recovery Feature, to set DULBE=1.
3. Perform a Get Feature command for FID 05h, Error Recovery Feature, to ensure that DULBE=1.
4. Configure the NVMe Host to issue a Write command to write a known data pattern to the controller.
5. Configure the NVMe Host to issue a Dataset Management command with the Attribute – Deallocate (AD) field set to '1' and specifying the same LBA range written to in the previous step.
6. Configure the NVMe Host to issue the following commands to the same LBA range deallocated in the previous step.
    a. Read
    b. Verify (if supported) (Check Bit 7 of ONCS field)

c. Compare (if supported) (Check bit 0 of ONCS field)

Observable Results:

1. Verify that the Read command fails with status Deallocated or Unwritten Logical Block, 87h.

Case 3: **Allocate via Write (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Perform a Set Feature command for FID 05h, Error Recovery Feature, to set DULBE=0.
3. Perform a Get Feature command for FID 05h, Error Recovery Feature, to ensure that DULBE=0.
4. Configure the NVMe Host to issue a Dataset Management command with the Attribute – Deallocate (AD) field set to '1' .
5. Perform an Identify Namespace Data Structure Command and record the value of the NUSE field.
6. Configure the NVMe Host to a Write command to the same LBA range deallocated in the previous steps.
7. Perform an Identify Namespace Data Structure Command and record the value of the NUSE field.

Observable Results:

1. Verify that the value reported in the NUSE field in the second Identify Namespace Data Structure Command is greater than the value reported in the first Identify Namespace Data Structure Command.

Case 4: **Allocate via Write Uncorrectable (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports Write Uncorrectable. If the command is not supported this test is not applicable.
3. Perform a Set Feature command for FID 05h, Error Recovery Feature, to set DULBE=0.
4. Perform a Get Feature command for FID 05h, Error Recovery Feature, to ensure that DULBE=0.
5. Configure the NVMe Host to issue a Dataset Management command with the Attribute – Deallocate (AD) field set to '1' and specifying the same LBA range written to in the previous step.
6. Perform an Identify Namespace Data Structure Command and record the value of the NUSE field.
7. Configure the NVMe Host to issue a Write Uncorrectable command to the same LBA range deallocated in the previous steps.
8. Perform an Identify Namespace Data Structure Command and record the value of the NUSE field.

Observable Results:

1. Verify that the value reported in the NUSE field in the second Identify Namespace Data Structure Command is greater than the value reported in the first Identify Namespace Data Structure Command.

Case 5: **Allocate via Write Zeroes (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports Dataset Management. If the command is not supported this test is not applicable.
2. Check the ONCS field of the Identify Controller Data structure to determine if the DUT supports Write Zeroes. If the command is not supported this test is not applicable.
3. Perform a Set Feature command for FID 05h, Error Recovery Feature, to set DULBE=0.
4. Perform a Get Feature command for FID 05h, Error Recovery Feature, to ensure that DULBE=0.
5. Configure the NVMe Host to issue a Dataset Management command with the Attribute – Deallocate (AD) field set to '1' and specifying the same LBA range written to in the previous step.
6. Perform an Identify Namespace Data Structure Command and record the value of the NUSE field.

7. Configure the NVMe Host to issue a Write Zeroes command to the same LBA range deallocated in the previous steps, with the Deallocate bit (CDW12.DEAC) set to 0.
8. Perform an Identify Namespace Data Structure Command and record the value of the NUSE field.

Observable Results:

1. Verify that the value reported in the NUSE field in the second Identify Namespace Data Structure Command is greater than the value reported in the first Identify Namespace Data Structure Command.

Possible Problems: None.

Group 1: NVM Features

Overview:

This section describes a method for performing conformance verification for NVMe products implementing NVM Features.

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 2.14 – Metadata Handling (M, OF-FYI)

Purpose: To verify that an NVMe Controller properly handles metadata.

References:

[1] NVMe Specification 8.2

Resource Requirements:

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

Last Modification: November 28, 2017

Discussion: Metadata is additional data allocated on a per logical block basis. There is no requirement for how the host makes use of the metadata area. One of the most common usages for metadata is to convey end–to–end protection information. The metadata may be transferred by the controller to or from the host in one of two ways. The mechanism used is selected when the namespace is formatted.

One of the transfer mechanisms shall be selected for each namespace when it is formatted; transferring a portion of metadata with one mechanism and a portion with the other mechanism is not supported.

Test Setup: See Appendix A.

Case 1: **Extended LBA (M, OF-FYI)**

The first mechanism for transferring the metadata is as a contiguous part of the logical block that it is associated with. The metadata is transferred at the end of the associated logical block, forming an extended logical block. In this case, both the logical block data and logical block metadata are pointed to by the PRP1 and PRP2 pointers (or SGL Entry 1 if SGLs are used).

Test Procedure:

1. Check the Metadata Size (MS) field in the Identify LBA Format Data Structure, which is within the Identify Namespace Data Structure. If MS is set to 0, then Metadata is not supported and this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller specifying an LBA in a namespace which has been formatted to allow metadata via extended LBAs with known data patterns for both the data and metadata.
3. Configure the NVMe Host to issue a Read command to the controller specifying the LBA which was previously written to.

Observable Results:

1. Verify that the data and metadata returned by the controller exactly match the data and metadata patterns which were written.

Case 2: **Separate Buffer (M, OF-FYI)**

The second mechanism for transferring the metadata is as a separate buffer of data. In this case, the metadata is pointed to with the Metadata Pointer, while the logical block data is pointed to by the Data Pointer. When a command uses PRPs for the metadata in the command, the metadata is required to be physically contiguous. When a command uses SGLs for the metadata in the command, the metadata is not required to be physically contiguous.

Test Procedure:

1. Check the Metadata Size (MS) field in the Identify LBA Format Data Structure, which is within the Identify Namespace Data Structure. If MS is set to 0, then Metadata is not supported and this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller specifying an LBA in a namespace which has been formatted to allow metadata via a separate buffer with known data patterns for both the data and metadata.

3. Configure the NVMe Host to issue a Read command to the controller specifying the LBA which was previously written to.

Observable Results:

1. Verify that the data and metadata returned by the controller exactly match the data and metadata patterns which were written.

Possible Problems: Support for each metadata transfer mechanism is indicated on a per namespace basis according to the FLBAS field Bit 4 of the Identify Namespace data structure. If there is no active namespaces which supports a specific transfer mechanism then tests utilizing that mechanism cannot be performed. If the NVMe Device targets NVMe Specification revision 1.2 or higher and the NVMe Controller Under Test supports the Namespace Management command, then the NVMe Host may create a new namespace which supports a specific transfer mechanism in order to perform procedure steps which require that transfer mechanism.

Test 2.15 – End–to–end Data Protection (M)

Purpose: To verify that an NVMe Controller can properly handle end–to–end data protection.

References:

[1] NVMe Specification 8.3

Resource Requirements:

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

Last Modification: March 2, 2016

Discussion: To provide robust data protection from the application to the NVM media and back to the application itself, end–to–end data protection may be used. If this optional mechanism is enabled, then additional protection information (e.g. CRC) is added to the LBA that may be evaluated by the controller and/or host software to determine the integrity of the logical block.

This additional protection information, if present, is either the first eight bytes of metadata or the last eight bytes of metadata, based on the format of the namespace. For metadata formats with more than eight bytes, if the protection information is contained within the first eight bytes of metadata, then the CRC does not cover any metadata bytes. For metadata formats with more than eight bytes, if the protection information is contained within the last eight bytes of metadata, then the CRC covers all metadata bytes up to but excluding these last eight bytes.

Test Setup: See Appendix A.

Case 1: **Write Command Processing (M)**

Protection information processing may occur as a side effect of a Write command. If the namespace was formatted with protection information and the PRACT bit is cleared to '0', then logical block data and metadata containing protection information are transferred from the host to NVM. As the logical block and metadata pass through the controller, the protection information is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End–to–end Guard Check, End–to–end Application Tag Check or End–to end Reference Tag Check). If the namespace was formatted with protection information and the PRACT bit is set to '1', then logical block data is transferred from the host to the controller. The controller inserts protection information and the logical block data and metadata containing protection information are written to NVM.

Test Procedure:

1. Check the DPC field of the Identify Namespace Data Structure to determine if the DUT supports End to End Data Protection Capabilities. If DPC set to 0, this test is not applicable.
2. Configure the NVMe Host to issue a Write command to the controller with the PRACT bit cleared to '0' and specifying an LBA in a namespace which has been formatted with protection information and also specifying properly formatted protection information with a known data pattern.
3. Configure the NVMe Host to issue a Write command to the controller with the PRACT bit cleared to '0' and specifying an LBA in a namespace which has been formatted with protection information and also specifying improperly formatted protection information.
4. Configure the NVMe Host to issue a Write command to the controller with the PRACT bit set to '1' and specifying an LBA in a namespace which has been formatted with protection information and also specifying a known data pattern.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. For steps 1 and 4, verify that the completion queue entry for each command indicates Success status.

3. For step 3, verify that the completion queue entry for the command indicates an appropriate status code (i.e., End–to–end Guard Check, End–to–end Application Tag Check or End–to end Reference Tag Check depending on how the protection information was improperly formatted).

Case 2: **Read Command Processing (M)**

Protection information processing may occur as a side effect of a Read command. The processing parallels Write command processing with the exception that logical block data flows in the opposite direction. When the PRACT bit is cleared to '0' and the namespace was formatted with protection information, logical block data and metadata are transferred from NVM to the host and checked by the controller. When the PRACT bit is set to '1' and the namespace was formatted with protection information, logical block data and metadata are transferred from the NVM to the controller. The controller checks the protection information and then removes it from the metadata before passing the LBA to the host. If the namespace format contains metadata beyond the protection information, then the protection information is not stripped regardless of the state of the PRACT bit (i.e., the metadata field remains the same size in the host as that in NVM).

Test Procedure:

1. Check the DPC field of the Identify Namespace Data Structure to determine if the DUT supports End to End Data Protection Capabilities. If DPC set to 0, this test is not applicable.
2. Configure the NVMe Host to issue a Read command to the controller with the PRACT bit cleared to '0' and specifying the LBA which was written to in Case 1 step 1.
3. Configure the NVMe Host to issue a Read command to the controller with the PRACT bit cleared to '0' and specifying the LBA which was written to in Case 1 step 4.
4. Configure the NVMe Host to issue a Write command to the controller with the PRACT bit set to '1' and specifying the LBA which was written to in Case 1 step 1.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.
2. For step 1, verify that data returned by the controller exactly matches the data and metadata which was written.
3. For step 3, verify that the data returned by the controller exactly matches the data which was written and contains additional metadata containing properly formatted and correct protection information.
4. For step 4, verify that the data returned by the controller contains no protection information.

Possible Problems:

Case 2 requires reading the data which was written in Case 1. Therefore these test cases must be performed in succession or the data must be rewritten as part of the Case 2 test procedure implementation.

Test 2.16 – Power Management (M, OF)

Purpose: To verify that an NVMe Controller can properly handle power management**.**

References:

[1] NVMe Specification 8.4

Resource Requirements:

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

Last Modification: June 24, 2015

Discussion: The power management capability allows the host to manage NVM subsystem power statically or dynamically. Static power management consists of the host determining the maximum power that may be allocated to an NVM subsystem and setting the NVM Express power state to one that consumes this amount of power or less. Dynamic power management consists of the host modifying the NVM Express power state to best satisfy changing power and performance objectives. This power management mechanism is meant to complement and not replace autonomous power management performed by a controller.

The number of power states implemented by a controller is returned in the Number of Power States Supported (NPSS) field in the Identify Controller data structure. A controller shall support at least one power state and may optionally support up to a total of 32 power states. Power states are contiguously numbered starting with zero such that each subsequent power state consumes less than or equal to the maximum power consumed in the previous state. Thus, power state zero indicates the maximum power that the NVM subsystem is capable of consuming.

Associated with each power state is a Power State Descriptor in the Identify Controller data structure. The descriptors for all implemented power states may be viewed as forming a table as shown in Table 6 which contains sample power state values for a controller with seven implemented power states.

Test Setup: See Appendix A.

Case 1: **Relative Write Latency (M, OF)**

Relative Write Latency (RWL): This field indicates the relative write latency associated with this power state. The value in this field shall be less than the number of supported power states (e.g., if the controller supports 16 power states, then valid values are 0 through 15). A lower value means lower write latency.

Table 6 – Example Power State Descriptor Table

Power State	Maximum Power (MP) (W)	Entry Latency (ENTLAT) (µs)	Exit Latency (EXLAT) (µs)	Relative Read Throughput (RRT)	Relative Read Latency (RRL)	Relative Write Throughput (RWT)	Relative Write Latency (RWL)
0	25	5	5	0	0	0	0
1	18	5	7	0	0	1	0
2	18	5	8	1	0	0	0
3	15	20	15	2	0	2	0
4	10	20	30	1	1	3	0
5	8	50	50	2	2	4	0
6	5	20	5000	4	3	5	1

Test Procedure:

1. Configure the NVMe Host to issue an Identify command specifying CNS value 01h to the controller in order to receive back an Identify Controller data structure.

2. For each power state supported by the controller as indicated by the NPSS field, read the RWL field of the associated Power State Descriptor.

Observable Results:
1. Verify that the relative write latency values are all less than the number of supported power states.

Case 2: **Relative Write Throughput (M, OF)**

Relative Write Throughput (RWT): This field indicates the relative write throughput associated with this power state. The value in this field shall be less than the number of supported power states (e.g., if the controller supports 16 power states, then valid values are 0 through 15). A lower value means higher write throughput.

The amount of data an application can write to stable storage on the server over a period of time is a measurement of the write throughput of a distributed file system. Write throughput is therefore an important aspect of performance

Test Procedure:
1. Configure the NVMe Host to issue an Identify command specifying CNS value 01h to the controller in order to receive back an Identify Controller data structure.
2. For each power state supported by the controller as indicated by the NPSS field, read the RWL field of the associated Power State Descriptor.

Observable Results:
1. Verify that the relative write throughput values are all less than the number of supported power states.

Case 3: **Relative Read Latency (M, OF)**

Relative Read Latency (RRL): This field indicates the relative read latency associated with this power state. The value in this field shall be less than the number of supported power states (e.g., if the controller supports 16 power states, then valid values are 0 through 15). A lower value means lower read latency.

Test Procedure:
1. Configure the NVMe Host to issue an Identify command specifying CNS value 01h to the controller in order to receive back an Identify Controller data structure.
2. For each power state supported by the controller as indicated by the NPSS field, read the RWL field of the associated Power State Descriptor.

Observable Results:
1. Verify that the relative read latency values are all less than the number of supported power states.

Case 4: **Relative Read Throughput (M, OF)**

Relative Read Throughput (RRT): This field indicates the relative read throughput associated with this power state. The value in this field shall be less than the number of supported power states (e.g., if the controller supports 16 power states, then valid values are 0 through 15). A lower value means higher read throughput.

Test Procedure:
1. Configure the NVMe Host to issue an Identify command specifying CNS value 01h to the controller in order to receive back an Identify Controller data structure.
2. For each power state supported by the controller as indicated by the NPSS field, read the RWL field of the associated Power State Descriptor.

Observable Results:
1. Verify that the relative read throughput values are all less than the number of supported power states.

Case 5: **Power Management Feature (M, OF)**

The host may dynamically modify the power state using the Set Features command and determine the current power state using the Get Features command. The host may directly transition between any two supported power states.

The Entry Latency (ENTLAT) field in the power management descriptor indicates the maximum amount of time in microseconds that it takes to enter that power state and the Exit Latency (EXLAT) field indicates the maximum amount of time in microseconds that it takes to exit that state. The maximum amount of time to transition between any two power states is equal to the sum of the old state's exit latency and the new state's entry latency. The host is not required to wait for a previously submitted power state transition to complete before initiating a new transition.

Test Procedure:

1. Configure the NVMe Host to issue a Set Features command for the Power Management Feature for each of the supported Power State values supported by the NVMe Controller to the controller.
2. After each Set Features command completes, configure the NVMe Host to issue a Get Features command for the Power Management Feature to the controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. For each Get Features command, verify that the controller successfully transitioned to the selected power state.

Possible Problems: None.

Test 2.17 – Host Memory Buffer (M)

Purpose: To verify that an NVMe system can properly handle supports host memory buffer.

References:

[1] NVMe Specification 8.9, 5.14.1.13, NVMe v1.3 ECN 004a

Resource Requirements:

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

Last Modification: April 29, 2019

Discussion: The Host Memory Buffer feature allows the controller to utilize an assigned portion of host memory exclusively. The use of the host memory resources is vendor specific. Host software may not be able to provide any or a limited amount of the host memory resources requested by the controller. The controller shall function properly without host memory resources. Refer to section 5.14.1.13 of the NVMe Specification.

During initialization, host software may provide a descriptor list that describes a set of host memory address ranges for exclusive use by the controller. The host memory resources assigned are for the exclusive use of the controller (host software should not modify the ranges) until host software requests that the controller release the ranges and the controller completes the Set Features command. The controller is responsible for initializing the host memory resources. Host software should request that the controller release the assigned ranges prior to a shutdown event, a Runtime D3 event, or any other event that requires host software to reclaim the assigned ranges. After the controller acknowledges that it is no longer using the ranges, host software may reclaim the host memory resources. In the case of Runtime D3, host software should provide the host memory resources to the controller again and inform the controller that the ranges were in use prior to the RTD3 event and have not been modified.

Test Setup: See Appendix A.

Case 1: **Proper Structure (M)**

Test Procedure:

1. Check the HMPRE field of the Identify Controller data structure to determine if the DUT supports Host Memory Buffer. If the HMPRE field is set to 0, this test is not applicable.
2. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Admin Completion Queue indicating the status for the command.
2. Verify that the Host Memory Buffer Preferred Size (HMPRE) field of the Identify Controller data structure is larger than or equal to the value indicated by the Host Memory Buffer Minimum Size (HMMIN) field.

Case 2: **Configuration (FYI)**

The Host Memory Feature controls the Host Memory Buffer. The attributes are indicated in Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15.

The Host Memory Buffer feature provides a mechanism for the host to allocate a portion of host memory for the controller to use exclusively. After a successful completion of a Set Features enabling the host memory buffer, the host shall not write to the associated host memory region, buffer size, or descriptor list until the host memory buffer has been disabled.

After a successful completion of a Set Features command that disables the host memory buffer, the

controller shall not access any data in the host memory buffer until the host memory buffer has been enabled. The controller should retrieve any necessary data from the host memory buffer in use before posting the completion queue entry for the Set Features command. Posting of the completion queue entry for the Set Features command acknowledges that it is safe for the host software to modify the host memory buffer contents. Refer to section 8.9 of the NVMe Specification.

Test Procedure:

1. Check the HMPRE field of the Identify Controller data structure to determine if the DUT supports Host Memory Buffer. If the HMPRE field is set to 0, this test is not applicable.
2. Configure the NVMe Host to issue a Set Features command for the Host Memory Buffer feature with the Enable Host Memory (EHM) bit set to '1' to the NVMe Controller.
3. Configure the NVMe Host to issue a Get Features command for the Host Memory Buffer feature to the NVMe Controller.
4. Configure the NVMe Host to issue a Set Features command for the Host Memory Buffer feature with the Enable Host Memory (EHM) bit cleared to '0' to the NVMe
5. Controller. Configure the NVMe Host to issue a Get Features command for the Host Memory Buffer feature to the NVMe Controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify in each case that the value for EHM returned by the controller in response to the Get Feature command matches what was set by the Host in the Set Feature command.

Case 3: **Reset Persistent (FYI)**

The host memory resources are not persistent in the controller across a reset event. Host software should provide the previously allocated host memory resources to the controller after the reset completes. If host software is providing previously allocated host memory resources (with the same contents) to the controller, the Memory Return bit is set to ‘1’ in the Set Features command. The controller shall ensure that there is no data loss or data corruption in the event of a surprise removal while the Host Memory Buffer feature is being utilized.

Test Procedure:

1. Check the HMPRE field of the Identify Controller data structure to determine if the DUT supports Host Memory Buffer. If the HMPRE field is set to 0, this test is not applicable.
2. Configure the NVMe Host to issue a Get Features command for the Host Memory Buffer feature to the NVMe Controller.
3. Configure the NVMe Host to issue a Set Features command for the Host Memory Buffer feature with the Enable Host Memory (EHM) bit set to '1' and Save (SV) set to 0, to the NVMe Controller.
4. Configure the NVMe Host to issue a Get Features command for the Host Memory Buffer feature to the NVMe Controller.
5. Configure the NVMe Host to issue a Controller Level Reset to the NVMe Controller.
6. Configure the NVMe Host to issue a Get Features command for the Host Memory Buffer feature to the NVMe Controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify that the DUT responds to the Get Features command in step 4 with EHM set to 1.
Verify that after the controller reset, the controller responds to the Get Feature command with EHM set to 0.

Case 4: **Enable HMB when Already Enabled (FYI)**

Test Procedure:

1. Check the HMPRE field of the Identify Controller data structure to determine if the DUT supports Host Memory Buffer. If the HMPRE field is set to 0, this test is not applicable.
2. Configure the NVMe Host to issue a Get Features command for the Host Memory Buffer feature to the NVMe Controller.
3. Configure the NVMe Host to issue a Set Features command for the Host Memory Buffer feature with the Enable Host Memory (EHM) bit set to '1' and Save (SV) set to 0, to the NVMe Controller.
4. Configure the NVMe Host to issue a Get Features command for the Host Memory Buffer feature to the NVMe Controller.
5. Configure the NVMe Host to issue a Set Features command for the Host Memory Buffer feature with the Enable Host Memory (EHM) bit set to '1' and Save (SV) set to 0, to the NVMe Controller.

Observable Results:

1. Verify that the DUT responds to the Set Features command in step 5 with status 'Command Sequence Error'.

Case 5: **Disable HMB when Already Disabled (FYI)**

Test Procedure:

1. Check the HMPRE field of the Identify Controller data structure to determine if the DUT supports Host Memory Buffer. If the HMPRE field is set to 0, this test is not applicable.
2. Configure the NVMe Host to issue a Get Features command for the Host Memory Buffer feature to the NVMe Controller.
3. Configure the NVMe Host to issue a Set Features command for the Host Memory Buffer feature with the Enable Host Memory (EHM) bit set to '1' and Save (SV) set to 0, to the NVMe Controller.
4. Configure the NVMe Host to issue a Get Features command for the Host Memory Buffer feature to the NVMe Controller.
5. Configure the NVMe Host to issue a Set Features command for the Host Memory Buffer feature with the Enable Host Memory (EHM) bit set to '0' and Save (SV) set to 0, to the NVMe Controller.
6. Configure the NVMe Host to issue a Get Features command for the Host Memory Buffer feature to the NVMe Controller.
7. Configure the NVMe Host to issue a Set Features command for the Host Memory Buffer feature with the Enable Host Memory (EHM) bit set to '0' and Save (SV) set to 0, to the NVMe Controller.

Observable Results:

1. Verify that the DUT responds to the Set Features command in step 7 with status Success.

Possible Problems: None.

Test 2.18 – Replay Protected Memory Block (IP)

Purpose: To verify that an NVMe system can properly handle replay protected memory blocks.

References:

[1] NVMe Specification 8.10

Resource Requirements:

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

Last Modification: April 9, 2019

Discussion: The Replay Protected Memory Block (RPMB) provides a means for the system to store data to a specific memory area in an authenticated and replay protected manner. This is provided by first programming authentication key information to the controller that is used as a shared secret. The system is not authenticated in this phase, therefore the authentication key programming should be done in a secure environment (e.g., as part of the manufacturing process). The authentication key is utilized to sign the read and write accesses made to the replay protected memory area with a Message Authentication Code (MAC). Use of random number (nonce) generation and a write count register provide additional protection against replay of messages where messages could be recorded and played back later by an attacker.

The controller may support multiple RPMB targets. RPMB targets are not contained within a namespace. Security Send and Security Receive commands for RPMB do not use the namespace ID field; NSID shall be cleared to 0h. Each RPMB target operates independently – there may be requests outstanding to multiple RPMB targets at once (where the requests may be interleaved between RPMB targets). In order to guarantee ordering the host should issue and wait for completion for one Security Send or Security Receive command at a time. Each RPMB target requires individual authentication and key programming. Each RPMB target may have its own unique Authentication Key.

The message types defined in Figure 223 of the NVMe Specification are used by the host to communicate with an RPMB target. Request Message Types are sent from the host to the controller. Response Message Types are sent to the host from the controller.

Test Setup: See Appendix A.

Case 1: **RPMB Operations (IP)**

The host sends a Request Message Type to the controller to request an operation by the controller or to deliver data to be written into the RPMB memory block. To deliver a Request Message Type, the host uses the Security Send command. If the data to be delivered to the controller is more than reported in Identify Controller data structure, the host sends multiple Security Send commands to transfer the entire data.

The host sends a Response Message Type to the controller to read the result of a previous operation request, to read the Write Counter, or to read data from the RPMB memory block. To deliver a Response Message Type, the host uses the Security Receive command. If the data to be read from the controller is more than reported in Identify Controller data structure, the host sends multiple Security Receive commands to transfer the entire data.

Test Procedure:

1. Check the RPMBS field in the Identify Controller Data Strucutre to determine if the controller supports RPMB. If RPMB is not supported this. Test is not applicable.
2. Configure the NVMe Host to issue a Security Send command with the feature ID set to Replay Protected Memory Blocks to the NVMe Controller.
3. Configure the NVMe Host to issue a Security Receive command with the feature ID set to Replay Protected Memory Blocks 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 data received is correct, and if the data is larger than reported in the Identify Controller data structure the host sends multiple Security Receive commands to transfer the entire data.

Case 2: **Authentication Key Programming (IP)**

Authentication Key programming is initiated by a Security Send command to program the Authentication Key to the specified RPMB target, followed by a subsequent Security Send command to request the result, and lastly, the host issues a Security Receive command to retrieve the result.

Test Procedure:

1. Check the RPMBS field in the Identify Controller Data Structure to determine if the controller supports RPMB. If RPMB is not supported this. Test is not applicable.
2. Configure the NVMe Host to issue a Security Send command for the Replay Protected Memory Blocks Feature to the NVMe Controller, with the RPMB target set to a target to access, and the Mac/Key field set to the key to be programmed, and the Request/Response field set to "0001h".
3. Configure the NVMe Host to issue a Security Send command for the Replay Protected Memory Blocks Feature to the NVMe Controller, with the RPMB target set to a target to access, and the Request/Response field set to "0005h".
4. Configure the NVMe Host to issue a Security Receive command for the Replay Protected Memory Blocks Feature to the NVMe Controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify the Security Receive command is populated correctly by the NVMe Controller.

Case 3: **Read Write Counter Value (IP)**

The Read Write Counter Value sequence is initiated by a Security Send command to request the Write Counter value, followed by a Security Receive command to retrieve the Write Counter result.

Test Procedure:

1. Check the RPMBS field in the Identify Controller Data Structure to determine if the controller supports RPMB. If RPMB is not supported this. Test is not applicable.
2. Configure the NVMe Host to issue a Security Send command for the Replay Protected Memory Blocks Feature to the NVMe Controller, with the RPMB target set to a target to access, and the Nonce field set to a specific Nonce generated by the host, and the Request/Response field set to "0002h".
3. Configure the NVMe Host to issue a Security Receive command for the Replay Protected Memory Blocks Feature to the NVMe Controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify the Security Receive command is populated correctly by the NVMe Controller.

Case 4: **Authenticated Data Write (IP)**

The success of programming the data should be checked by the host by reading the result register of the RPMB.

1. Check the RPMBS field in the Identify Controller Data Structure to determine if the controller supports RPMB. If RPMB is not supported this. Test is not applicable.
2. The host initiates the Authenticated Data Write verification process by issuing a Security Send command with delivery of a RPMB data frame containing the Request Message Type = 0005h.

3. The controller returns a successful completion of the Security Send command when the verification result is ready for retrieval.
4. The host should then retrieve the verification result by issuing a Security Receive command.
5. The controller returns a successful completion of the Security Receive command and returns the RPMB data frame containing the Response Message Type = 0300h, the incremented counter value, the data address, the MAC and result of the data programming operation.

Test Procedure:

1. Configure the NVMe Host to issue a Security Send command for the Replay Protected Memory Blocks Feature to the NVMe Controller, with the Mac/Key field set to a MAC generated by the host, and the RPMB target set to a target to access, and the Write Counter field set to the current write counter value, and the Address field set to the RPMB address, and the Sector Count field set the to the number of 512B blocks, and the Request/Response field set to "0003h", and the Data field set to data to be written.
2. Configure the NVMe Host to issue a Security Send command for the Replay Protected Memory Blocks Feature to the NVMe Controller, with the RPMB target set to a target to access, and the Request/Response field set to "0005h"
3. Configure the NVMe Host to issue a Security Receive command for the Replay Protected Memory Blocks Feature to the NVMe Controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify the Security Receive command is populated correctly by the NVMe Controller.

Case 5: **Authenticated Data Read (IP)**

The Authenticated Data Read sequence is initiated by a Security Send command. The RPMB data frame delivered from the host to the controller includes the Request Message Type = 0004h, Nonce, Address, and the Sector Count.

When the controller receives this RPMB Data Frame, it first checks the Address. If there is an error in the Address then the result is set to 0004h (address failure) and the data read is not valid. When the host receives a successful completion of the Security Send command from the controller, it should send a Security Receive command to the controller to retrieve the data. The controller returns an RPMB Data Frame with Response Message Type (0400h), the Sector Count, a copy of the Nonce received in the request, the Address, the Data, the controller calculated MAC, and the Result. Note: It is the responsibility of the host to verify the MAC returned on an Authenticated Data Read Request.

If the data transfer from the addressed location in the controller fails, the returned Result is 0006h (read failure). If the Address provided in the Security Send command is not valid, then the returned Result is 0004h (address failure). If another error occurs during the read procedure then the returned Result is 0001h (general failure).

Test Procedure:

1. Check the RPMBS field in the Identify Controller Data Structure to determine if the controller supports RPMB. If RPMB is not supported this. Test is not applicable.
2. Configure the NVMe Host to issue a Security Send command for the Replay Protected Memory Blocks Feature to the NVMe Controller, with the RPMB target set to a target to access, and the Nonce field set to a Nonce generated by the host, and the Address field set to the RPMB address, and the Sector Count set to the number of 512B blocks, and the Request/Response field set to "0004h", and the Data field set to data to be written.
3. Configure the NVMe Host to issue a Security Receive command for the Replay Protected Memory Blocks Feature to the NVMe Controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify the Security Receive command is populated correctly by the NVMe Controller.

Possible Problems: This is a Mandatory if Supported test, depending on device feature support according to the Replay Protected Memory Block Support (RPMBS) field of the Identify Controller data structure.

Test 2.19 – IO Determinism (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller can properly supports predictable latency mode if supported.

References:

[1] NVMe Specification TP 4003c

Resource Requirements:

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

Last Modification: April 22, 2019

Discussion: Predictable Latency Mode is used to achieve predictable latency for read and write operations. When configured to operate in this mode using the Predictable Latency Mode Config Feature, the namespaces in an NVM Set provide windows of operation for deterministic operation or non- deterministic operation. Support for predictable latency mode also requires support for associated Features and Log Pages.

Test Setup: See Appendix A.

Case 1: **Predictable Latency Mode Supported (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 5 of the CTRATT field in the Identify Controller Data Structure. If CTRATT Bit 5 is set to 0, this test is not applicable.
2. Configure the Testing Station to issue a Set Feature for Feature Identifier 13h “Predictable Latency Mode Config” to enable Predictable Latency Mode.
3. Configure the Testing Station to issue a Get Feature for Feature Identifier 13h “Predictable Latency Mode Config”.
4. Configure the Testing Station to issue a Get Feature for Feature Identifier 14h “Predictable Latency Mode Window”.
5. Configure the Testing Station to issue a Get Log Page for Log Identifier 0Ah “Predictable Latency Per NVM Set”.
6. Configure the Testing Station to issue a Get Log Page for Log Identifier 0Bh “Predictable Latency Event Aggregate”.
7. Configure the Testing Station to issue a Get Feature for Feature Identifier 13h “Predictable Latency Mode Config”.

Observable Results:

1. Verify that each of the Get Log and Get Feature operations completed with status “Success”.

Case 2: **Predictable Latency Mode Not Supported (M, OF)**

Test Procedure:

1. Check Bit 5 of the CTRATT field in the Identify Controller Data Structure. If CTRATT Bit 5 is set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Set Feature for Feature Identifier 13h “Predictable Latency Mode Config” to enable Predictable Latency Mode.
3. Configure the Testing Station to issue a Get Feature for Feature Identifier 13h “Predictable Latency Mode Config”.
4. Configure the Testing Station to issue a Get Feature for Feature Identifier 14h “Predictable Latency Mode Window”.
5. Configure the Testing Station to issue a Get Log Page for Log Identifier 0Ah “Predictable Latency Per NVM Set”.

6. Configure the Testing Station to issue a Get Log Page for Log Identifier 0Bh "Predictable Latency Event Aggregate".
7. Configure the Testing Station to issue a Get Feature for Feature Identifier 13h "Predictable Latency Mode Config".

Observable Results:

1. Verify that None of the Get Log and Get Feature operations completed with status "Success".

Case 3: **Predictable Latency Mode Not Enabled (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 5 of the CTRATT field in the Identify Controller Data Structure. If CTRATT Bit 5 is set to 1, this test is not applicable.
2. Configure the Testing Station to issue a Get Feature for Feature Identifier 13h "Predictable Latency Mode Config".
3. If Predictable Latency Mode is enabled, configure the Testing Station to issue a Set Feature for Feature Identifier 13h "Predictable Latency Mode Config" to disable Predictable Latency Mode. Otherwise, proceed to the next step.
4. Configure the Testing Station to issue a Get Feature for Feature Identifier 14h "Predictable Latency Mode Window".

Observable Results:

1. Verify that the Get Feature command for FID 14h returned an error of "Invalid Field in Command".

Possible Problems: None known.

Test 2.20 – Namespace Write Protection (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller has properly implemented Namespace Write Protection if supported.

References:

[1] NVMe Specification TP 4005b

Resource Requirements:

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

Last Modification: April 24, 2019

Discussion: Namespace Write Protection is a optional Feature identifier that controls write protection on a per namespace basis. This Feature may be used to prevent modification of the specified namespace. The controller fails commands that attempt to modify the namespace while it is write protected.

Test Setup: See Appendix A.

Case 1: **Enable and Disable Write Protection (FYI, OF)**

Test Procedure:

1. Check Namespace Write Protection Capabilities (NWPC) in the Identify Controller Data Structure. If Bits 0, 1, 2 of NWPC are set to 0, the DUT does not support Namespace Write Protection, and this test is not applicable.
2. Perform a Get Feature Operation for the Namespace Write Protection Config (Feature ID 84h). Verify that Write Protection State is 000b. If not, perform a Set Feature operation to set Namespace Write Protection Config to 'No Write Protect' 000b. If the protection state of the namespace is set to Write Protect Until Power Cycle (010b) or Permanent Write Protect (011b), then the Set Feature command will fail with an error of Feature Not Changeable and this test is not applicable.
3. Perform a Write operation to write a known data pattern (i.e. AAAAh) to the Namespace.
4. Perform a Set Feature operation to set Namespace Write Protection Config to 'Write Protect' 001b.
5. Perform the following Admin commands to the namespace.
    a. Device Self Test
    b. Get Feature
    c. Get Log
    d. Identify
6. Perform the following NVM commands to the namespace.
    a. Compare
    b. Read
    c. Flush
    d. Verify
7. Perform a Write command with a known data pattern which is different than the data pattern used in Step 3 to the namespace (i.e. BBBBh).
8. Perform a Read command to the namespace.
9. Perform a Set Feature operation to set Namespace Write Protection Config to 'No Write Protect' 000b.
10. Perform a Write command with a known data pattern, different than the data pattern used in Step 3 to the namespace (i.e. CCCCh).
11. Perform a Read command to the namespace.

Observable Results:

1. Verify that each of the commands in step 5 and 6 completed successfully.
2. Verify that the Read operation performed in Step 8, returned the data pattern written in Step 2, and not the data pattern attempted to be written in Step 7.
3. Verify that the Read operation performed in Step 11, returned the data pattern written in Step 10.

4. Verify that the Write operation attempted in Step 7 completed with status “Namespace is Write Protected” (20h).

Possible Problems: None known.

Test 2.21 – Persistent Memory Region (M, OF)

Purpose: To verify that an NVMe Controller has properly implemented Persistent Memory Region if supported.

References:

[1] NVMe Specification 4.8

Resource Requirements:

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

Last Modification: May 21, 2020

Discussion: The Persistent Memory Region (PMR) is an optional region of general purpose PCI Express read/write persistent memory that may be used for a variety of purposes. The controller indicates support for the PMR by setting CAP.PMRS to '1' and indicates whether the controller supports command data and metadata transfers to or from the PMR by setting support flags in the PMRCAP register. When command data and metadata transfers to or from PMR are supported, all data and metadata associated with a particular command shall be either entirely located in the Persistent Memory Region or outside the Persistent Memory Region.

Test Setup: See Appendix A.

Case 1: **PMR Persistence Across Reset (M, OF)**

Test Procedure:

1. Check that CAP.PMRS is set to 1. If CAP.PMRS is not set to 1, the DUT does not support Persistent Memory Region and this test is not applicable.
2. The Testing Station acting as a Host should set PMRMSC.CBA to a valid value.
3. Enable the PMR controller memory space via PMRMSC.CMSE and PMRCTL.EN.
4. Wait 2x the timer specified in PMRCAP.PMRTO
5. Check that the controller indicates that the PMR is ready by clearing PMRSTS.NRDY to 0.
6. Perform a WRITE operation of a non-zero known pattern to the address range specified for the PMR.
7. Perform a Conventional Reset, PCIe Hot Reset..
8. Wait 2x the timer specified in PMRCAP.PMRTO
9. Perform a READ operation to the address range specified for the PMR.
10. Perform a WRITE operation of all 0s' to the address range specified for the PMR.
11. Perform a READ operation to the address range specified for the PMR.

Observable Results:

1. Verify that each READ operation returned the same known data pattern that was previously written to the PMR.

Case 2: **PMR Persistence Across Disable and Enable (M, OF)**

Test Procedure:

1. Check that CAP.PMRS is set to 1. If CAP.PMRS is not set to 1, the DUT does not support Persistent Memory Region and this test is not applicable.
2. The Testing Station acting as a Host should set PMRMSC.CBA to a valid value.
3. Enable the PMR controller memory space via PMRMSC.CMSE and PMRCTL.EN.
4. Wait 2x the timer specified in PMRCAP.PMRTO
5. Check that the controller indicates that the PMR is ready by clearing PMRSTS.NRDY to 0.
6. Perform a WRITE operation of a non-zero known pattern to the address range specified for the PMR.
7. Disable the PMR using the PMRCTL.EN bit.
8. Wait 2x the timer specified in PMRCAP.PMRTO

9. Enable the PMR using the PMRCTL.EN bit.
10. Wait 2x the timer specified in PMRCAP.PMRTO
11. Perform a READ operation to the address range specified for the PMR.
12. Perform a WRITE operation of all 0s' to the address range specified for the PMR.
13. Perform a READ operation to the address range specified for the PMR.

Observable Results:

1. Verify that each READ operation returned the same known data pattern that was previously written to the PMR.

Possible Problems: None known.

Test 2.22 – Rebuild Assist via Get LBA Status (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller has properly implemented Get LBA Status functionality if supported.

References:

[1] NVMe Specification 5.27, 8.22

Resource Requirements:

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

Last Modification: May 4, 2020

Discussion: Potentially Unrecoverable LBAs are LBAs that, when read, may result in the command that caused the media to be read being aborted with Unrecovered Read Error status. The Get LBA Status capability provides the host with the ability to identify Potentially Unrecoverable LBAs. The logical block data is able to be recovered from another location and re-written.

Test Setup: See Appendix A.

Case 1: **Get LBA Status (FYI, OF-FYI)**

Test Procedure:

1. Check that Bit 9 of the OACS field in the Identify Controller Data Structure is set to 1, to indicate support for the Get LBA Status capability. If this bit is not set to 1, this test is not applicable.
2. Check that Bit 13 of the OAES field in the Identify Controller Data Structure is set to 1, to indicate support for LBA Status Notification event. If this bit is not set to 1, this test is not applicable.
3. Perform a Get Log Page command for the LBA Status Information log page (0Eh).
4. Perform a Get Feature command for the LBA Status Attributes feature (15h).
5. Perform a Set Feature command for the LBA Status Attributes feature (15h) to set a new value for the LBA Status Information Report Interval (LSIRI).
6. Perform a Get Feature command for the LBA Status Attributes feature (15h).
7. Perform a Get LBA Status command with ATYPE set to 10h.
8. Perform a Get LBA Status command with ATYPE set to 11h.

Observable Results:

1. Verify that the controller indicates support for the Log Page Offset and extended Numbre of Dwords (i.e. 32 bits rather than 12 bits in the Log Page Attributes field of the Idenitfy Controller data structure.
2. Verify that the Get Log Page operation for the LBA Status Information log page (0Eh) completes successfully.
3. Verify that each Get Feature operation for the LBA Status Attributes feature (15h) completes successfully.
4. Verify that the Get Feature operation following the Set Feature operation for feature 15h indicates the value set by the host.
5. Verify that the Get LBA Status command with ATYPE=0 completes successfully and includes a properly formatted LBA Status Descriptor list and:
    a. the controller returns Untracked LBAs and Tracked LBAs in the range specified in the Get LBA Status command for the namespace specified in the Namespace Identifier (CDW1.NSID);
    b. the controller removes all LBAs in the range specified in the Get LBA Status command, which prior to processing the Get LBA Status command were successfully re-written, from relevant internal data structures (e.g., internal Tracked LBA list).
6. Verify that the Get LBA Status command with ATYPE=1 completes successfully and includes a properly formatted LBA Status Descriptor list and:
    a. returns Tracked LBAs in the range specified in the Get LBA Status command for the namespace specified in the Namespace Identifier (CDW1.NSID) field;

b. removes all LBAs in the range specified in the Get LBA Status command, which prior to processing the Get LBA Status command were successfully re-written, from relevant internal data structures (e.g., internal Tracked LBA list);

Possible Problems: Since the controller identifies which LBAs to track, the polling internal of tracked LBA status is not tested, as there is not a deterministic way to cause LBAs to be tracked.

Test 2.23 – Improved Performance Parameters (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller operates correctly even when advertised improved performance parameters are not used by the host.

References:

[1] NVMe Specification 5.15, 8.25

Resource Requirements:

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

Last Modification: May 4, 2020

Discussion: NVMe controllers may require constrained I/O sizes and alignments to achieve the full performance potential. There are a number of optional attributes that the controller uses to indicate these recommendations. If hosts do not follow these constraints, then the controller shall function correctly, but performance may be limited.

Test Setup: See Appendix A.

Case 1: **Unaligned Write Command (FYI, OF-FYI)**

Test Procedure:

1. Check the Identify Namespace Data Structure NSFEAT bit 4. If this bit is not set to 1, then this test is not applicable.
2. Check the Identify Namespace Data Structure and record the following values:
    a. Namespace Preferred Write Granularity (NPWG)
    b. Namespace Preferred Write Alignment (NPWA)
    c. Namespace Preferred Deallocate Granularity (NPDG)
    d. Namespace Preferred Deallocate Alignment NPDA
3. Perform a Direction Operation for Return Parameters (01h) and record the following values.
    a. Stream Write Size (SWS)
    b. Stream Granularity Size (SGS)
4. Perform a Write command which addresses a number of logical blocks (NLB) which is not a multiple of NPWG and SWS and the SLBA field is not aligned to NPWA.

Observable Results:

1. Verify that Write operation completes successfully.

Case 2: **Unaligned Write Uncorrectable Command (FYI, OF-FYI)**

Test Procedure:

1. Check the Identify Namespace Data Structure NSFEAT bit 4. If this bit is not set to 1, then this test is not applicable.
2. Check the ONCS field in the Identify Controller Data Structure to determine if the DUT supports the Write Uncorrectable command. If this command is not supported this test is not applicable.
3. Check the Identify Namespace Data Structure and record the following values:
    a. Namespace Preferred Write Granularity (NPWG)
    b. Namespace Preferred Write Alignment (NPWA)
    c. Namespace Preferred Deallocate Granularity (NPDG)
    d. Namespace Preferred Deallocate Alignment NPDA
4. Perform a Direction Operation for Return Parameters (01h) and record the following values.
    a. Stream Write Size (SWS)
    b. Stream Granularity Size (SGS)

5. Perform a Write Uncorrectable command which addresses a number of logical blocks (NLB) which is not a multiple of NPWG and SWS and the SLBA field is not aligned to NPWA.

Observable Results:
1. Verify that Write Uncorrectable operation completes successfully.

Case 3: **Unaligned Write Zeroes Command (FYI, OF-FYI)**

Test Procedure:
1. Check the Identify Namespace Data Structure NSFEAT bit 4. If this bit is not set to 1, then this test is not applicable.
2. Check the ONCS field in the Identify Controller Data Structure to determine if the DUT supports the Write Zeroes command. If this command is not supported this test is not applicable.
3. Check the Identify Namespace Data Structure and record the following values:
    a. Namespace Preferred Write Granularity (NPWG)
    b. Namespace Preferred Write Alignment (NPWA)
    c. Namespace Preferred Deallocate Granularity (NPDG)
    d. Namespace Preferred Deallocate Alignment NPDA
4. Perform a Direction Operation for Return Parameters (01h) and record the following values.
    a. Stream Write Size (SWS)
    b. Stream Granularity Size (SGS)
5. Perform a Write Zeroes command which addresses a number of logical blocks (NLB) which is not a multiple of NPWG and SWS and the SLBA field is not aligned to NPWA.

Observable Results:
1. Verify that Write Zeroes operation completes successfully.

Case 4: **Unaligned Dataset Management Command (FYI, OF-FYI)**

Test Procedure:
1. Check the Identify Namespace Data Structure NSFEAT bit 4. If this bit is not set to 1, then this test is not applicable.
2. Check the ONCS field in the Identify Controller Data Structure to determine if the DUT supports the Dataset Management command. If this command is not supported this test is not applicable.
3. Check the Identify Namespace Data Structure and record the following values:
    a. Namespace Preferred Write Granularity (NPWG)
    b. Namespace Preferred Write Alignment (NPWA)
    c. Namespace Preferred Deallocate Granularity (NPDG)
    d. Namespace Preferred Deallocate Alignment NPDA
4. Perform a Direction Operation for Return Parameters (01h) and record the following values.
    a. Stream Write Size (SWS)
    b. Stream Granularity Size (SGS)
5. Perform a Dataset Management command, with AD bit set to 1, which addresses a number of logical blocks (NLB) which is not a multiple of NPDG, and the start of each range is not aligned to NPDA and SGS logical blocks.

Observable Results:
1. Verify that Dataset Management operation completes successfully.

Possible Problems: None known.

Test 2.24 – Controller Memory Buffer (M, OF)

Purpose: To verify that an NVMe Controller correctly implements controller memory buffer if supported.

References:

[1] NVMe Specification 4.7

Resource Requirements:

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

Last Modification: May 18, 2020

Discussion: The Controller Memory Buffer (CMB) is a region of general purpose read/write memory on the controller. The controller indicates support for the CMB by setting CAP.CMBS to '1'. The host indicates intent to use the CMB by setting CMBMSC.CRE to '1'. Once this bit is set to '1', the controller indicates the properties of the CMB via the CMBLOC and CMBSZ registers.

Test Setup: See Appendix A.

Case 1: **CMB Supported (M, OF)**

Test Procedure:

1. Check CAP.CMBS. If CAP.CMBS=0, then this test is not applicable.
2. The Testing Station acting as a Host should set CMBMSC.CBA to a valid value.
3. The Testing Station acting as a Host should set CMBMSC.CRE to 1.
4. Read the CMBLOC and CMBSZ registers.
5. Perform a WRITE operation of a known data pattern to the CMB using the parameters in CMBLOC and CMBSZ.
6. Perform a READ operation to the CMB using the parameters in CMBLOC and CMBSZ.

Observable Results:

1. Verify that WRITE operation completes successfully.
2. Verify that READ operation completes successfully and that the DUT returns the same data written in the previous WRITE operation.

Possible Problems: None known.

Test 2.25 – NVM Sets (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller correctly implements NVM Sets if supported.

References:

[1] NVMe Specification 4.9

Resource Requirements:

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

Last Modification: December 1, 2019

Discussion: An NVM Set is a collection of NVM that is separate (logically and potentially physically) from NVM in other NVM Sets. One or more namespaces may be created within an NVM Set and those namespaces inherit the attributes of the NVM Set. A namespace is wholly contained within a single NVM Set and shall not span more than one NVM Set.

Test Setup: See Appendix A.

Case 1: **NVM Sets Supported (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 2 (NVM Sets) of the CTRATT field of the Identify Controller Data Structure. If Bit 2 is not set to 1 , then this test is not applicable.
2. Record the value of Bit 4 (Endurance Groups) of the CTRATT field of the Identify Controller Data Structure.
3. Perform an Identify Command to the DUT using CNS 00h (Identify Namespace Data Structure).
4. Perform an Identify Command to the DUT using CNS 04h (NVM Set List).
5. Perform a Namespace Management command with operation Create Namespace using the NVM Set Identifier.

Observable Results:

1. Verify that the DUT returns an NVM Set List with properly formatted NVM Set Attributes Entries in response to the Identify Command to CNS 04h, including the associated Endurance Group.
2. Verify that the DUT indicates the NVM Set Identifier with which the namespace is associate in the response to the Identify Command to CNS 00h.
3. Verify that the DUT indicates support for Endurance Groups by checking Bit 4 (Endurance Groups) of the CTRATT field of the Identify Controller Data Structure.
4. Verify that the Namespace Management command completes successfully.

Case 2: **RLL Supported in NVM Set (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 2 (NVM Sets) of the CTRATT field of the Identify Controller Data Structure. If Bit 2 is not set to 1, then this test is not applicable.
2. Check Bit 3 (Read Recovery Level) of the CTRATT field of the Identify Controller Data Structure. If Bit 3 is not set to 1 , then this test is not applicable.
3. Record the value of Bit 4 (Endurance Groups) of the CTRATT field of the Identify Controller Data Structure.
4. Perform an Identify Command to the DUT using CNS 00h (Identify Namespace Data Structure).
5. Perform an Identify Command to the DUT using CNS 04h (NVM Set List).
6. Perform a Namespace Management command with operation Create Namespace using the NVM Set Identifier.
7. Perform a Set Feature command for the Read Recovery Level Feature for a specific NVM Set Identifier.

Observable Results:

1. Verify that the DUT returns an NVM Set List with properly formatted NVM Set Attributes Entries in response to the Identify Command to CNS 04h, including the associated Endurance Group.
2. Verify that the DUT indicates the NVM Set Identifier with which the namespace is associate in the response to the Identify Command to CNS 00h.
3. Verify that the DUT indicates support for Endurance Groups by checking Bit 4 (Endurance Groups) of the CTRATT field of the Identify Controller Data Structure.
4. Verify that the Namespace Management command completes successfully.
5. Verify that the Set Feature command for the Read Recovery Level Feature completes successfully.

Case 3: **Endurance Group Info Log Matches SMART/Health Log Summary (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Controller Data Structure Command (CNS=01h).
2. Check Bit 2 (NVM Sets) of the CTRATT field of the Identify Controller Data Structure. If Bit 2 is not set to 1, then this test is not applicable.
3. Check Bit 4 (Endurance Groups) of the CTRATT field of the Identify Controller Data Structure. If Bit 4 is not set to 1, then this test is not applicable.
4. Perform an Identify Namespace Data Structure Command (CNS=00h) for all attached Namespaces. Record the Endurance Group Identifier (ENDGID) for the given Namespaces.
5. Perform Get Log Page for the Endurance Group Info Log (09h). Repeat this for all recorded Endurance Group Identifiers.
6. Perform Get Log Page for the SMART/Health Log (02h) for the Controller.

Observable Results:

1. Verify that any bits set to 1 in the Critical Warning field of any retrieved Endurance Group Info Log Pages are also set to 1 in the Endurance Group Critical Warning Summary field of the SMART/Health Log Page.

Case 4: **Endurance Group Identifier = 0 (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Controller Data Structure Command (CNS=01h).
2. Check Bit 2 (NVM Sets) of the CTRATT field of the Identify Controller Data Structure. If Bit 2 is not set to 1, then this test is not applicable.
3. Check Bit 4 (Endurance Groups) of the CTRATT field of the Identify Controller Data Structure. If Bit 4 is not set to 1, then this test is not applicable.
4. Perform an Identify Namespace Data Structure Command (CNS=00h) for all attached Namespaces. Record the Endurance Group Identifier (ENDGID) for the given Namespaces.
5. Perform a Get Log Page for the Endurance Group Info Log (09h) using an Endurance Group Identifier of 0h.

Observable Results:

1. Verify that the Get Log Page command is aborted with status ‘Invalid Field in Command’.

Case 5: **Endurance Group Identifier Does not Exist for Set Feature (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Controller Data Structure Command (CNS=01h).
2. Check Bit 2 (NVM Sets) of the CTRATT field of the Identify Controller Data Structure. If Bit 2 is not set to 1, then this test is not applicable.
3. Check Bit 4 (Endurance Groups) of the CTRATT field of the Identify Controller Data Structure. If Bit 4 is not set to 1, then this test is not applicable.
4. Perform an Identify Namespace Data Structure Command (CNS=00h) for all attached Namespaces. Record the Endurance Group Identifier (ENDGID) for the given Namespaces.

5. Perform a Set Feature command for the Endurance Group Event Configuration (18h) using an Endurance Group Identifier that does not exist.

Observable Results:
1. Verify that the Set Feature command is aborted with status 'Invalid Field in Command'.

Case 6: **Endurance Group Identifier Does not Exist for Get Feature (FYI, OF-FYI)**

Test Procedure:
1. Perform an Identify Controller Data Structure Command (CNS=01h).
2. Check Bit 2 (NVM Sets) of the CTRATT field of the Identify Controller Data Structure. If Bit 2 is not set to 1, then this test is not applicable.
3. Check Bit 4 (Endurance Groups) of the CTRATT field of the Identify Controller Data Structure. If Bit 4 is not set to 1, then this test is not applicable.
4. Perform an Identify Namespace Data Structure Command (CNS=00h) for all attached Namespaces. Record the ENGIDMAX for the given Namespaces.
5. Perform a Get Feature command for the Endurance Group Event Configuration (18h) using an Endurance Group Identifier = ENGIDMAX + 1 .

Observable Results:
1. Verify that the Get Feature command is aborted with status 'Invalid Field in Command'.

Case 7: **Endurance Group Critical Warnings Configuration with Reserved Field (FYI, OF-FYI)**

Test Procedure:
1. Perform an Identify Controller Data Structure Command (CNS=01h).
2. Check Bit 2 (NVM Sets) of the CTRATT field of the Identify Controller Data Structure. If Bit 2 is not set to 1, then this test is not applicable.
3. Check Bit 4 (Endurance Groups) of the CTRATT field of the Identify Controller Data Structure. If Bit 4 is not set to 1, then this test is not applicable.
4. Perform an Identify Namespace Data Structure Command (CNS=00h) for all attached Namespaces. Record the Endurance Group Identifier (ENDGID) for the given Namespaces.
5. Perform Set Feature command for the Endurance Group Event Configuration (18h) using a valid Endurance Group Identifier, and Bit 23 set to 1. Bit 23 should correspond to Bit 7 in the Critical Warning Field of the Endurance Group Log, which is reserved in NVMe specification v1.4.

Observable Results:
1. Verify that the Set Feature command is aborted with status 'Invalid Field in Command'.

Case 8: **Endurance Groups not Supported (FYI, OF-FYI)**

Test Procedure:
1. Perform an Identify Controller Data Structure Command (CNS=01h).
2. Check Bit 4 (Endurance Groups) of the CTRATT field of the Identify Controller Data Structure. If Bit 4 is set to 1, then this test is not applicable.
3. Perform a Set feature Command for the Asynchronous Event Configuration Feature (FID=0Bh) with the Endurance Group Event Aggregate Log Change Notices bit set to 1.

Observable Results:
1. Verify that the Set Feature command is aborted with status 'Invalid Field in Command'.

Possible Problems: None known.

Test 2.26 – Read Recovery Level (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller correctly implements NVM Sets if supported.

References:

[1] NVMe Specification 8.16

Resource Requirements:

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

Last Modification: May 4, 2020

Discussion: The Read Recovery Level (RRL) is a NVM Set configurable attribute that balances the completion time for read commands and the amount of error recovery applied to those read commands. The Read Recovery Level applies to an NVM Set with which the Read Recovery Level is associated. A namespace created within an NVM Set inherits the Read Recovery Level of that NVM Set. If NVM Sets are not supported, all namespaces in the NVM subsystem use an identical Read Recovery Level.

Test Setup: See Appendix A.

Case 1: **Proper RLL Levels Supported (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Controller Data Structure command.
2. Check Bit 3 (Read Recovery Level) of the CTRATT field of the Identify Controller Data Structure. If Bit 3 is not set to 1, then this test is not applicable.
3. Perform a WRITE operation to a given namespace writing a known Data pattern.
4. Perform a READ operation to the same namespace as was written in the previous step, ensuring that the known pattern was written.
5. Perform a Get Feature operation to FID 12h to query the current Read Recovery Level in the same namespace.
6. Perform a Set Feature operation to FID 12h change the Read Recovery Level to another supported value, in the same namespace, as indicated in the Read Recovery Levels Supported field in the Identify Controller data structure.
7. Perform a Get Feature operation to FID 12h to query the current Read Recovery Level, verifying that the previously supplied value is returned.
8. Perform a READ operation to the same namespace, ensuring that the previously written known pattern is still present.

Observable Results:

1. Verify the DUT indicates support for at least Read Recovery Levels 4 and 15 in the Read Recovery Levels Supported field in the Identify Controller data structure.

Case 2: **NVM Sets Not Supported (FYI, OF-FYI)**

Test Procedure:

1. Check Bit 3 (Read Recovery Level) of the CTRATT field of the Identify Controller Data Structure. If Bit 3 is not set to 1 , then this test is not applicable.
2. Record the RRLS value in the Identify Controller Data Structure.
3. Check Bit 2 (NVM Sets) of the CTRATT field of the Identify Controller Data Structure. If Bit 2 is set to 1 (NVM Sets supported), then this test is not applicable.
4. Perform a Get Feature Command to FID 12h to retrieve the support RRL.

Observable Results:

1. Verify that the RRL value returned in response to the Get Feature command matched the RRLS values reported in the Identify Controller Data Structure.

Possible Problems: None known.

Test 2.27 – Asymmetric Namespace Access Reporting (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller correctly implements Asymmetric Namespace Access if supported.

References:

[1] NVMe Specification 5.14.1.12, 8.20

Resource Requirements:

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

Last Modification: December 1, 2019

Discussion: Asymmetric Namespace Access (ANA) occurs in environments where namespace access characteristics (e.g., performance or ability to access the media) may vary based on the controller used to access the namespace (e.g., Fabrics) and the internal configuration of the NVM subsystem. Asymmetric Namespace Access Reporting is used to indicate to the host information about those access characteristics.

Test Setup: See Appendix A.

Case 1: **ANA Log Page RGO=0 (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Controller Data Structure command. Record Bit 3 of the CMIC field. If Bit 3 is set to 0, this test is not applicable.
2. Perform a Get Log Page for Log Page ID 0Ch with the RGO bit set to 0.

Observable Results:

1. Verify that the Get Log Page command for Log Page ID 0Ch completed successfully.
2. Verify that the returned LPOL and LPOU fields are cleared to 0.
3. Verify that returned ANA Group Descriptors contain the Namespace Identifiers of attached namespaces that are members of the ANA Group described by that ANA Group Descriptor and the Number of NSID Values field set to the number of Namespace Identifier values in that ANA Group Descriptor.

Case 2: **ANA Log Page RGO=1 (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Controller Data Structure command. Record Bit 3 of the CMIC field. If Bit 3 is set to 0, this test is not applicable.
2. Perform a Get Log Page for Log Page ID 0Ch with the RGO bit set to 1.

Observable Results:

1. Verify that the Get Log Page command for Log Page ID 0Ch completed successfully, and that Log Page returned included ANA Group Descriptors with the Number of NSID value field in each ANA Group Descriptor cleared to 0h.

Possible Problems: None known.

Test 2.28 – Namespace Granularity (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller correctly implements Namespace Granularity if supported.

References:

[1] NVMe Specification 8.12

Resource Requirements:

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

Last Modification: December 1, 2019

Discussion: The size granularity and the capacity granularity are hints which may be used by the host to minimize the capacity that is allocated for a namespace and that is not able to be addressed by logical block addresses. The granularities are used in specifying values for the Namespace Size (NSZE) and Namespace Capacity (NCAP) fields of the data structure used for the create operation of the Namespace Management command.

Test Setup: See Appendix A.

Case 1: **Create Namespace with all capacity Allocated (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure CTRATT Bit 7 to determine if the product under test supports Namespace Granularity. If the product under test does not support Namespace Granularity (Bit 7 = 1) this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 16h and a valid value for Controller ID in CDW10.CNTID.
3. Perform a Namespace Management command to Create a namespace with the following conditions
    a. The Product of NSZE and the Formatted LBA Size is an integral multiple of the Namespace Size Granularity.
    b. The Product of NCAP and the Formatted LBA Size is an integral multiple of the Namespace Size Granularity.
    c. NSZE is equal to NCAP.
4. Attached the Namespace created in the previous step.
5. Perform at least 1 WRITE and READ operation to the Namespace.
6. Detach and Delete the namespace.

Observable Results:

1. Verify that the Identify command for CNS=16h completes successfully and includes a properly formatted Namespace Granularity List.
2. Verify that the namespace granularity descriptors with an index greater than the value in the Number of Descriptors field are cleared to 0h.
3. Verify that the Namespace Management Create command completes successfully.
4. Verify that the WRITE, READ, Namespace Detach, and Namespace Delete commands complete successfully.

Case 2: **Create Namespace with not all capacity Allocated (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure CTRATT Bit 7 to determine if the product under test supports Namespace Granularity. If the product under test does not support Namespace Granularity (Bit 7 = 1) this test is not applicable.

2. Configure the NVMe Host to issue an Identify command specifying CNS value 16h and a valid value for Controller ID in CDW10.CNTID.
3. Perform a Namespace Management command to Create a namespace with the following conditions
    a. The Product of NSZE and the Formatted LBA Size is not an integral multiple of the Namespace Size Granularity.
    b. The Product of NCAP and the Formatted LBA Size is not an integral multiple of the Namespace Size Granularity.
    c. NSZE is not equal to NCAP.
4. Attached the Namespace created in the previous step.
5. Perform at least 1 WRITE and READ operation to the Namespace.
6. Detach and Delete the namespace.

Observable Results:

1. Verify that the Identify command for CNS=16h completes successfully and includes a properly formatted Namespace Granularity List.
2. Verify that the namespace granularity descriptors with an index greater than the value in the Number of Descriptors field are cleared to 0h.
3. Verify that the Namespace Management Create command completes successfully.
4. Verify that the WRITE, READ, Namespace Detach, and Namespace Delete commands complete successfully.

Possible Problems: None known.

Test 2.29 – SQ Associations (FYI, OF-FYI)

Purpose: To verify that an NVMe Controller correctly implements SQ Associations if supported.

References:

[1] NVMe Specification 8.23

Resource Requirements:

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

Last Modification: December 1, 2019

Discussion: The SQ Associations capability provides hints to the controller as to which specific I/O Queues are associated with a given NVM Set. The controller uses this information to further enhance performance when Predictable Latency Mode is enabled. A controller which supports SQ Associations shall also support NVM Sets and Predictable Latency Mode.

Test Setup: See Appendix A.

Case 1: **Associated Features Supported (FYI, OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure CTRATT Bit 8 to determine if the product under test supports SQ Associations. If the product under test does not support SQ Associations (Bit 8 = 1) this test is not applicable.

Observable Results:

1. Verify that CTRATT Bit 2 (NVM Sets) is set to 1.
2. Verify that CTRATT Bit 5 (Predictable Latency Mode) is set to 1.

Possible Problems: None known.

Test 2.30 – Transport SGL Data Block Descriptor (OF-FYI)

Purpose: To verify that an NVMe Controller correctly implements SGL Data Block Descriptor if supported.

References:

[1] NVMe Specification 4.2, 4.4, 5.15

Resource Requirements:

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

Last Modification: June 4, 2020

Discussion: A controller may support byte or dword alignment and granularity of Data Blocks. If a controller supports only dword alignment and granularity as indicated in the SGL Support field of the Identify Controller data structure, then the values in the Address and Length fields of all Data Block descriptors shall have their lower two bits cleared to 00b. This requirement applies to Data Block descriptors that indicate data and/or metadata memory regions. This test only applies to devices using a Fabrics transport.

Test Setup: See Appendix A.

Case 1: **Correct Descriptor Format (OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure SGLS Bit 21 to determine if the product under test supports the Transport SGL Data Block Descriptor. If the product under test does not support the Transport SGL Data Block Descriptor (Bit 21 = 1) this test is not applicable.
2. Record Bits 01:00 of the SGLS field.
    a. If this field equals 01b. then SGLs are supported, and there are no alignment or granularity requirement for Data Blocks.
    b. If this field equals 10b then SGLs are supported, and there is a dword alignment and granularity requirement for Data Blocks.
3. Perform an NVMe Connect command with a value valid for the transport being used.
    a. RDMA: Sub type values Ah to Eh are reserved, value Fh is used with the Keyed SGL Data Block Descriptor Type (4h).
    b. FC: Ah is the only value.
    c. TCP: 1h and Ah are the only valid values.

Observable Results:

1. Verify that if bits 01:00 of the SGLS field is set to 10b (DUT has dword alignment and granularity requirements), then the lower 2 bits of the Transport SGL Data Block Descriptor Length field are cleared to 00b.
2. Verify that the SGL Descriptor Type Field returned in the Transport SGL Data Block Descriptor is not set to a reserved value.
3. Verify that the connect command completed successfully.

Case 2: **Incorrect Descriptor Format (OF-FYI)**

Test Procedure:

1. Perform an Identify Command to CNS=01h, then check the Identify Controller Data Structure SGLS Bit 21 to determine if the product under test supports the Transport SGL Data Block Descriptor. If the product under test does not support the Transport SGL Data Block Descriptor (Bit 21 = 1) this test is not applicable.
2. Record Bits 01:00 of the SGLS field.

   a. If this field equals 01b. then SGLs are supported, and there are no alignment or granularity requirement for Data Blocks.
   b. If this field equals 10b then SGLs are supported, and there is a dword alignment and granularity requirement for Data Blocks.
3. Perform an NVMe Connect command with a value which is invalid for the transport being used.
   a. RDMA: Sub type values Ah to Eh are reserved, value Fh is used with the Keyed SGL Data Block Descriptor Type (4h).
   b. FC: Ah is the only value.
   c. TCP: 1h and Ah are the only valid values.

Observable Results:
1. Verify that the connect command did not complete successfully.

Possible Problems: This test is only described for NVMe-oF products.

Group 2: Controller Registers

Overview:

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

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 2.31 – Offset 00h: CAP – Memory Page Size Maximum (MPSMAX) (M, OF)

Purpose: To validate the MPSMAX field of the Controller Capabilities (CAP) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.1

Resource Requirements:

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

Last Modification: June 24, 2015

Discussion: This field indicates the maximum host memory page size that the controller supports. The maximum memory page size is (2 ^ (12 + MPSMAX)) (i.e. $2^{12+MPSMAX}$). Therefore, the maximum memory page size which a controller may support is 128MB. The host shall not configure a memory page size in CC.MPS that is larger than this value.

Test Setup: See.

Test Procedure:

1. Configure the NVMe Host to read the CAP.MPSMAX register field (bits 55:52) of the NVMe Controller.

Observable Results:

1. Verify that the value of the CAP.MPSMAX register field is greater than or equal to the value of the CAP.MPSMIN register field.

Possible Problems: None

Test 2.32 – Offset 00h: CAP – Memory Page Size Minimum (MPSMIN) (M, OF)

Purpose: To validate the MPSMIN field of the Controller Capabilities (CAP) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.1

Resource Requirements:

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

Last Modification: June 24, 2015

Discussion: This field indicates the minimum host memory page size that the controller supports. The minimum memory page size is (2 ^ (12 + MPSMIN)) (i.e. $2^{12+MPSMIN}$). Therefore, the minimum memory page size which a controller may support is 4KB. The host shall not configure a memory page size in CC.MPS that is smaller than this value.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CAP.MPSMIN register field (bits 51:48) of the NVMe Controller.

Observable Results:

1. Verify that the value of the CAP.MPSMAX register field is greater than or equal to the value of the CAP.MPSMIN register field.

Possible Problems: None**.**

Test 2.33 – Offset 00h: CAP – Command Sets Supported (CSS) (M, OF)

Purpose: To validate the Command Sets Supported (CSS) field of the Controller Capabilities (CAP) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.1

Resource Requirements:

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

Last Modification: January 4, 2021

Discussion: This field indicates the I/O Command Set(s) that the controller supports. A minimum of one command set shall be supported. The field is bit significant (bit definitions can be found in Table 7). If a bit is set to '1', then the corresponding I/O Command Set is supported. If a bit is cleared to '0', then the corresponding I/O Command Set is not supported.

Table 7 – Command Sets Supported Bit Definitions

Bit	Definition
37	NVM command set
38	Reserved
39	Reserved
40	Reserved
41	Reserved
42	Reserved
43	Controller Supports one or more I/O Command Sets.
44	No I/O Command Set is Supported.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CAP.CSS register field (bits 37:44) of the NVMe Controller.

Observable Results:

1. Verify that Bit 37 of the CAP.CSS register field is set to '1' to indicate that the NVM Command Set is supported by the controller . Controllers that support the NVM Command Set shall set this bit even if bit 43 is set to '1'.

Possible Problems: None.

Test 2.34 – Offset 00h: CAP – Doorbell Stride (DSTRD) (M, OF)

Purpose: To validate the Doorbell Stride (DSTRD) field of the Controller Capabilities (CAP) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.1

Resource Requirements:

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

Last Modification: March 2, 2016

Discussion: Each Submission Queue and Completion Queue Doorbell register is 32–bits in size. This register indicates the stride between doorbell registers. The stride is specified as (2 ^ (2 + DSTRD)) (i.e. $2^{2+DSTRD}$) in bytes. A value of 0h indicates a stride of 4 bytes, where the doorbell registers are packed without reserved space between each register.

Since there is no means to validate the value stored in this register field, this test is designed purely to determine the value the NVMe Controller returns when the register is read, and is therefore considered an informative test.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CAP.DSTRD register field (bits 35:32) of the NVMe Controller.
2. Report the value of the CAP.DSTRD register field.

Observable Results: None.

Possible Problems: There are no required values for the CAP.DSTRD register field at any given time and so this test is run as informative with no pass/fail criteria.

Test 2.35 – Offset 00h: CAP – Timeout (TO) (M, OF)

Purpose: To validate the Timeout (TO) field of the Controller Capabilities (CAP) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.1

Resource Requirements:

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

Last Modification: June 24, 2015

Discussion: –This is the worst case time that host software shall wait for CSTS.RDY to transition from:

a) '0' to '1' after CC.EN transitions from '0' to '1'; or
b) '1' to '0' after CC.EN transitions from '1' to '0'

This worst case time may be experienced after events such as an abrupt shutdown or activation of a new firmware image; typical times are expected to be much shorter. This field is in 500 millisecond units.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CAP.TO register field (bits 31:24) of the NVMe Controller.
2. Configure the NVMe Host to clear the CC.EN register field of the NVMe Controller to '0' to initiate a controller reset.
3. Configure the NVMe Host to query the CSTS.RDY register field every 100 milliseconds until it transitions from '1' to '0'.
4. Configure the NVMe Host to set the CC.EN register field of the NVMe Controller to '1' to re–enable the controller (be sure to set required registers field prior to re-enabling the controller).
5. Configure the NVMe Host to query the CSTS.RDY register field every 100 milliseconds until it transitions from '0' to '1'.

Observable Results:

1. Determine the elapsed time it takes for the CSTS.RDY register field value to transition after toggling the value of the CC.EN field. Verify that this value is less than or equal to the timeout value calculated from the CAP.TO register field read from the NVMe Controller.

Possible Problems: None

Test 2.36 – Offset 00h: CAP – Arbitration Mechanism Supported (AMS)(M, OF)

Purpose: To validate the Arbitration Mechanism Supported (AMS) field of the Controller Capabilities (CAP) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.1

Resource Requirements:

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

Last Modification: June 24, 2015

Discussion: This field is bit significant (bit definitions can be found in Table 8) and indicates the optional arbitration mechanisms supported by the controller. If a bit is set to '1', then the corresponding arbitration mechanism is supported by the controller. Refer to section 4.11 of the NVMe Specification for arbitration details.

Table 8 – CAP.AMS Bit Definitions

Bit	Definition:
17	Weighted Round Robin with Urgent Priority Class
18	Vendor Specific

The round robin arbitration mechanism is not listed since all controller shall support this arbitration mechanism.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CAP.AMS register field (bits 18:17) of the NVMe Controller.
2. For each arbitration mechanism supported by the NVMe Controller as indicated by the CAP.AMS register field, configure the NVMe Host to set the CC.AMS register field to the corresponding value.

Observable Results:

1. For each supported arbitration mechanism, verify that the NVMe Host is able to successfully set the CC.AMS register field to the corresponding value.

Possible Problems: The weighted Round Robin with Urgent and Vendor Specific arbitration mechanisms are optional to support. Therefore, it is valid for both bits of the CAP.AMS field to be cleared to '0'.

Test 2.37 – Offset 00h: CAP – Contiguous Queues Required (CQR) (M, OF)

Purpose: To validate the Contiguous Queues Required (CQR) field of the Controller Capabilities (CAP) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.1

Resource Requirements:

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

Last Modification: March 2, 2016

Discussion: This field is set to '1' if the controller requires that I/O Submission Queues and I/O Completion Queues are required to be physically contiguous. This field is cleared to '0' if the controller supports I/O Submission Queues and I/O Completion Queues that are not physically contiguous. If this field is set to '1', then the Physically Contiguous bit (CDW11.PC) in the Create I/O Submission Queue and Create I/O Completion Queue commands shall be set to '1'.

Since there is no means to validate the value stored in this register field, this test is designed purely to determine the value the NVMe Controller returns when the register is read, and is therefore considered an informative test.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CAP.CQR register field (bit 16) of the NVMe Controller.
2. Report the value of the CAP.CQR register field.

Observable Results: None.

Possible Problems: There are no required values for the CAP.CQR register field at any given time and so this test is run as informative with no pass/fail criteria.

Test 2.38 – Offset 00h: CAP – Maximum Queue Entries Supported (MQES) (M, OF)

Purpose: To validate the Maximum Queue Entries Supported (MQES) field of the Controller Capabilities (CAP) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.1

Resource Requirements:

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

Last Modification: June 24, 2015

Discussion: This field indicates the maximum individual queue size that the controller supports. This value applies to each of the I/O Submission Queues and I/O Completion Queues that host software may create. This is a 0's based value. The minimum value is 1h, indicating two entries.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CAP.MQES register field (bits 15:00) of the NVMe Controller.

Observable Results:

1. Verify that the value of the CAP.MQES register field is 1h or greater indicating 2 or more queue entries.

Possible Problems: None.

Test 2.39 – Offset 0Ch–10h: INTMS – Interrupt Mask Set and INTMC – Interrupt Mask Clear (M, OF)

Purpose: To validate the Interrupt Mask Set (INTMS) and Interrupt Mask Clear (INTMC) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.3, 3.1.4

Resource Requirements:

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

Last Modification: June 24, 2015

Discussion: The INTMS register is used to mask interrupts when using pin–based interrupts, single message MSI, or multiple message MSI. When using MSI–X, the interrupt mask table defined as part of MSI–X should be used to mask interrupts. Host software shall not access this register when configured for MSI–X; any accesses when configured for MSI–X is undefined.

The INTMC register is used to unmask interrupts when using pin–based interrupts, single message MSI, or multiple message MSI. When using MSI–X, then interrupt mask table defined as part of MSI–X should be used to unmask interrupts. Host software shall not access this register when configure for MSI–X; any accesses when configured for MSI–X is undefined.

The INTMS and INTMC registers both have a single field each: the Interrupt Vector Mask Set (IVMS) and Interrupt Vector Mask Clear (IVMC) fields respectively. Both fields are bit significant. If a '1' is written to a bit in the IVMS field, then the corresponding interrupt vector is masked from generating an interrupt or reporting a pending interrupt in the MSI Capability Structure. If a '1' is written to a bit in the IVMC field, then the corresponding interrupt vector is unmasked. Writing a '0' to a bit has no effect. When read, these fields return the current interrupt mask value within the controller (not the value of their register). If a bit has a value of '1', then the corresponding interrupt vector is masked. If a bit has a value of '0', then the corresponding interrupt vector is not masked.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the INTMS and INTMC registers of the NVMe Controller.
2. Configure the NVMe Host to write all zeros to both the INTMS and INTMC registers of the NVMe Controller.
3. Configure the NVMe Host to read the INTMS and INTMC registers of the NVMe Controller once again.

Observable Results:

1. Verify that the values returned by the registers do not change on subsequent reads after writing zeroes to their bits.

Possible Problems: None

Test 2.40 – Offset 14h: CC – I/O Completions Queue Entry Size (IOCQES) (M, OF)

Purpose: To validate the I/O Completion Queue Entry Size (IOCQES) field of the Controller Configuration (CC) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.5.

Resource Requirements:

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

Last Modification: April 2, 2019

Discussion: This field defines the I/O Completion Queue entry size that is used for the selected I/O Command Set. The required and maximum values for this field are specified in the Identify Controller data structure for each I/O Command Set. The value is in bytes and is specified as a power of two (2^*n*) (i.e. 2^n).

Test Setup: See Appendix A.

Test Procedure:

1. Check Identify Controller Data Structure for required minimum and maximum values of IOCQES.
2. Configure the NVMe Host to read the CC.IOCQES register field (bits 23:20) of the NVMe Controller.

Observable Results:

1. Verify that the maximum value for IOCQES advertised in the Identify Controller Data Structure is greater than or equal to the required minimum value
2. Verify that the value of the CC.IOCQES register field falls within the range specified for IOCQES as determined by the maximum and minimum values advertised in the Identify Controller Data Structure.

Possible Problems: None

Test 2.41 – Offset 14h: CC – I/O Submission Queue Entry Size (IOSQES) (M, OF)

Purpose: To validate the I/O Submission Queue Entry Size (IOCQES) field of the Controller Configuration (CC) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.5.

Resource Requirements:

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

Last Modification: April 2, 2019

Discussion: This field defines the I/O Submission Queue entry size that is used for the selected I/O Command Set. The required and maximum values for this field are specified in the Identify Controller data structure for each I/O Command Set. The value is in bytes and is specified as a power of two (2^*n*) (i.e. 2^n).

Test Setup: See Appendix A.

Test Procedure:

1. Check Identify Controller Data Structure for required minimum and maximum values of IOSQES.
2. Configure the NVMe Host to read the CC.IOSQES register field (bits 23:20) of the NVMe Controller.

Observable Results:

1. Verify that the maximum value for IOSQES advertised in the Identify Controller Data Structure is greater than or equal to the required minimum value
2. Verify that the value of the CC.IOSQES register field falls within the range specified for IOSQES as determined by the maximum and minimum values advertised in the Identify Controller Data Structure.

Possible Problems: None.

Test 2.42 – Offset 14h: CC – Shutdown Notification (SHN) (M, OF)

Purpose: To validate the Shutdown Notification (SHN) field of the Controller Configuration (CC) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.5.

Resource Requirements:

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

Last Modification: November 26, 2018

Discussion: This field is used to initiate shutdown processing when a shutdown is occurring, (i.e., a power down condition is expected). For a normal shutdown notification, it is expected that the controller is given time to process the shutdown notification. For an abrupt shutdown notification, the host may not wait for shutdown processing to complete before power is lost. The shutdown notification values are defined in Table 9.

Table 9 – CC.SHN Field Values

Value	Definition
00b	No notification; no effect.
01b	Normal shutdown notification.
10b	Abrupt shutdown notification.
11b	Reserved.

This field should be written by host software prior to any power down condition and prior to any change of the PCI power management state. It is recommended that this field also be written prior to a warm reboot. To determine when shutdown processing is complete, refer to CSTS.SHST.

It is recommended that the host wait a minimum of the RTD3 Entry Latency reported in the Identify Controller data structure for the shutdown operations to complete; if the value reported in RTD3 Entry Latency is 0h, then the host should wait for a minimum of one second. It is not recommended to disable the controller via the CC.EN field. This causes a Controller Reset which may impact the time required to complete shutdown processes.

It is safe to power off the controller when CSTS.SHST indicates shutdown processing is complete. It remains safe to power off the controller until CC.EN transitions from ‘0’ to ‘1’. To start executing commands on the controller after a shutdown operation, a Controller Reset is required. This initializat5ion sequence should then be executed.

Test Setup: See Appendix A.

Test Procedure:

1. Stop submitting any new I/O commands to the DUT and allow any outstanding commands to complete.
2. Perform a Delete I/O Submission Queue command to all existing I/O Submission queues.
3. Perform a Delete I/O Completion Queue command to all existing I/O Completion queues.
4. Configure the NVMe Host to read the CC.SHN register field (bits 15:14) of the NVMe Controller.
5. Configure the NVMe Host to write a value of 01b to the CC.SHN register field in order to initiate shutdown processing with a normal shutdown notification.
6. After the CSTS.SHST register field value transitions back to a value of 10b (shutdown processing complete), configure the NVMe Host to perform a full Controller Reset.
7. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h.
8. Configure the NVMe Host to write a value of 10b to the CC.SHN register field in order to initiate shutdown processing with an abrupt shutdown notification.
9. After the CSTS.SHST register field value transitions back to a value of 10b (shutdown processing complete), configure the NVMe Host to perform a full Controller Reset.

10. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h.

Observable Results:

1. Verify that the value of the CC.SHN register field is set to its default value of 00b (no notification) after a Controller Level Reset.
2. After the NVMe Host sets the value of the CC.SHN register field, verify that the controller sets the CSTS.SHST register field to its appropriate value as indicated in Table 12.
3. Verify that, after the NVMe Host performs a Controller Reset while CSTS.SHST indicates normal operation, the NVMe Host is able to issue commands to the NVMe Controller.

Possible Problems: None

Test 2.43 – Offset 14h: CC – Arbitration Mechanism Selected (AMS) (M, OF)

Purpose: To validate the Arbitration Mechanism Selected (AMS) field of the Controller Configuration (CC) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.5.

Resource Requirements:

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

Last Modification: February 1, 2018

Discussion: This field selects the arbitration mechanism to be used. This value shall only be changed when EN is cleared to '0'. Host software shall only set this field to supported arbitration mechanisms indicated in CAP.AMS. If this field is set to an unsupported value, the behavior is undefined. The arbitration mechanism values are defined in Table 10.

Table 10 – CC.AMS Field Values

Value	Definition
000b	Round Robin
001b	Weighted Round Robin with Urgent
010b – 110b	Reserved
111b	Vendor Specific.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to set CC.EN to 0.
2. Configure the NVMe Host to read the CC.AMS register field of the NVMe Controller.
3. Configure the NVMe Host to read the CAP.AMS register field of the NVMe Controller.
4. For each optional arbitration mechanism reported to be supported by the CAP.AMS register field:
    a. Configure the NVMe Host to set the CC.AMS register field to the appropriate value for that mechanism.
    b. Configure the NVMe Host to read the CC.AMS register field of the NVMe Controller.

Observable Results:

1. Verify that the value of the CC.AMS register field is set to its default value of 000b when CC.EN is set to 0.
2. Verify that the NVMe Controller properly allows the NVMe Host to set supported values in the CC.AMS register field.

Possible Problems: The weighted Round Robin with Urgent and Vendor Specific arbitration mechanisms are optional to support. Therefore, it is valid for both bits of the CAP.AMS field to be cleared to '0'.

Test 2.44 – Offset 14h: CC – I/O Command Set Selected (CSS) (M, OF)

Purpose: To validate the I/O Command set Selected (CSS) field of the Controller Configuration (CC) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.5.

Resource Requirements:

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

Last Modification: March 2, 2016

Discussion: This field specifies the I/O Command Set that is selected for use for the I/O Submission Queues. Host software shall only select a supported I/O Command Set, as indicated in CAP.CSS. This field shall only be changed when the controller is disabled (CC.EN is cleared to '0'). The I/O Command Set selected shall be used for all I/O Submission Queues. The command set values are defined in Table 11.

Table 11 – CC.CSS Field Values

Value	Definition
000b	NVM Command Set
001b – 111b	Reserved

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CC.CSS register field (bits 06:04) of the NVMe Controller.
2. For each I/O Command Set supported by the NVMe Controller as indicated by the CAP.CSS register field:
    a. Configure the NVMe Host to set the CC.CSS register field to the appropriate value for that I/O Command Set.
    b. Configure the NVMe Host to read the CC.CSS register field of the NVMe Controller.

Observable Results:

1. Verify that the value of the CC.CSS register field is set to its default value of 000b after the NVMe Host clears the CC.EN register field to '0' in order to initiate a Controller Reset (the host must set the CC.CSS field to a valid value prior to re-enabling the controller).
2. Verify that the NVMe Controller properly allows the NVMe Host to set supported values in the CC.CSS field.

Possible Problems: None

Test 2.45 – Offset 14h: CC – Enable (EN) (M, OF)

Purpose: To validate the Enable (EN) field of the Controller Configuration (CC) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.5.

Resource Requirements:

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

Last Modification: June 21, 2012

Discussion: When set to '1', the controller shall process commands based on Submission Queue Tail doorbell writes. When cleared to '0', the controller shall not process commands nor post completion queue entries to Completion Queues. When this field transitions from '1' to '0', the controller is reset (referred to as a Controller Reset). The reset deletes all I/O Submission Queues and I/O Completion Queues, resets the Admin Submission Queue and Completion Queue, and brings the hardware to an idle state. The reset does not affect PCI Express registers nor the Admin Queue registers (AQA, ASQ, or ACQ). All other controller registers defined in section 3 of the NVMe specification and internal controller state (e.g., Feature values defined in section 5 of the NVMe specification that are not persistent across power states) are reset to their default values. The controller shall ensure that there is no data loss for commands that have had corresponding completion queue entries posted to an I/O Completion Queue prior to the reset operation. Refer to section 7.3 for reset details.

When this field is cleared to '0', the CSTS.RDY bit is cleared to '0' by the controller once the controller is ready to be re–enabled. When this field is set to '1', the controller sets CSTS.RDY to '1' when it is ready to process commands. CSTS.RDY may be set to '1' before namespace(s) are ready to be accessed.

Setting this field from a '0' to a '1' when CSTS.RDY is a '1', or setting this field from a '1' to a '0' when CSTS.RDY is a '0' has undefined results. The Admin Queue registers (AQA, ASQ, and ACQ) shall only be modified when this field is cleared to '0'.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CC.EN register field (bit 00) of the NVMe Controller.

Observable Results:

1. Verify that when Bit 0 of the CC.EN register field is set to '1', the controller processes commands based on Submission Queue Tail doorbell writes.
2. Verify that when Bit 0 of the CC.EN register field is cleared to '0', the controller does not process commands nor post completion queue entries to Completion Queues.

Possible Problems: None

Test 2.46 – Offset 1Ch: CSTS – Shutdown Status (SHST) (M, OF)

Purpose: To validate the Shutdown Status (SHST) field of the Controller Status (CSTS) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.6.

Resource Requirements:

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

Last Modification: March 2, 2016

Discussion: This field indicates the status of shutdown processing that is initiated by the host setting the CC.SHN register field. The shutdown status values are defined in Table 12.

Table 12 – CSTS.SHST Field Values

Value	Definition
00b	Normal operation (no shutdown has been requested)
01b	Shutdown processing occurring
10b	Shutdown processing complete
11b	Reserved

To start executing commands on the controller after a shutdown operation (CSTS.SHST set to 10b), a reset (CC.EN cleared to '0') is required. If host software submits commands to the controller without issuing a reset, the behavior is undefined.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CSTS.SHST register field (bits 03:02) of the NVMe Controller.
2. Configure the NVMe Host to write a value of 01b to the CC.SHN register field in order to initiate shutdown processing with a normal shutdown notification.
3. Configure the NVMe Host to read the CSTS.SHST register field of the NVMe Controller.
4. After the CSTS.SHST register field value transitions back to a value of 10b (shutdown processing complete), configure the NVMe Host to perform a full Controller Reset.
5. Configure the NVMe Host to read the CSTS.SHST register field of the NVMe Controller.
6. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h.
7. Configure the NVMe Host to write a value of 10b to the CC.SHN register field in order to initiate shutdown processing with an abrupt shutdown notification.
8. Configure the NVMe Host to read the CSTS.SHST register field of the NVMe Controller.
9. After the CSTS.SHST register field value transitions back to a value of 10b (shutdown processing complete), configure the NVMe Host to perform a Controller Reset.
10. Configure the NVMe Host to read the CSTS.SHST register field of the NVMe Controller.
11. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h.

Observable Results:

1. Verify that the value of the CSTS.SHST register field is set to its default value of 00b (normal operation) after a Controller Level Reset.
2. After the NVMe Host sets the value of the CC.SHN register field to either 01b or 10b, verify that the NVMe Controller sets the CSTS.SHST register field to 01b to indicate that shutdown processing is occurring.
3. Record the time it takes for the NVMe Controller to transition the value of the CSTS.SHST register field from 01b to 10b (i.e. the shutdown processing time) for both normal and abrupt shutdown notifications.

4. Verify that, after the NVMe Host performs a Controller Reset while the CSTS.SHST register field indicates normal operation, the NVMe Host is able to issue commands to the NVMe Controller.

Possible Problems: None

Test 2.47 – Offset 1Ch: CSTS – Controller Fatal Status (CFS) (M, OF)

Purpose: To validate the Controller Fatal Status (CFS) field of the Controller Status (CSTS) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.6.

Resource Requirements:

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

Last Modification: March 2, 2016

Discussion: This field is set to ’1’ when a fatal controller error occurred that could not be communicated in the appropriate Completion Queue. This field is cleared to ‘0’ when a fatal controller error has not occurred. Refer to section 9.5.

The reset value of this field is ‘1’ when a fatal controller error is detected during controller initialization.

Since there is no means for a host to trigger a controller fatal status, there is no means to validate the value stored in this register field, and so this test is designed purely to determine the value the NVMe Controller returns when the register is read, and is therefore considered an informative test.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the CSTS.CFS field (bit 01) of the NVMe Controller.
2. Report the value of the CSTS.CFS field.

Observable Results: None.

Possible Problems: There are no explicit conditions for which the NVMe Host can generate a controller fatal status. Therefore this test is run as informative with no pass/fail criteria.

Test 2.48 – Offset -08h: CAP – Version (VS) (M, OF)

Purpose: To validate the Version (VS) field of the Capabilities (CAP) register of an NVMe Controller.

References:

[1] NVMe Specification 3.1.2

Resource Requirements:

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

Last Modification: November 14, 2017

Discussion: This register indicates the major and minor version of the NVM Express specification that the controller implementation supports. Valid versions of the specification are: 1.0, 1.1, and 1.2.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to read the VS.CAP field of the NVMe Controller.
2. Report the value of the VS.CAP field.
3. Configure the NVMe Host to issue an Identify command to the NVMe Controller specifying CNS value 01h in order to retrieve the Identify Controller data structure.

Report the value of rthe VER field in the Identify Controller data structure returned by the DUT.

Observable Results:

1. Verify that the DUT reports a valid value for VS. Valid values are 1.0, 1.1, 1.2, 1.3
2. Verify that the version reported in the Identify Controller data structure matches the value reported in the VS. CAP field.

Group 3: System Memory Structure

Overview:

This section describes a method for performing conformance verification for NVMe products implementing the NVMe System Memory Structure.

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 2.49 – Page Base Address and Offset (PBAO) (M, OF)

Purpose: To validate the Page Base Address and Offset field of PRP entries issued to an NVMe Controller.

References:

NVMe Specification 4.3

Resource Requirements:

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

Last Modification: February 1, 2018

Discussion: A physical region page (PRP) entry is a point to a physical memory page. PRPs are used as a scatter/gather mechanism for data transfers between the controller and memory. To enable efficient out of order data transfers between the controller and the host, PRP entries are a fixed size.

The size of the physical memory page is configured by host software in CC.MPS. shows the layout of a PRP entry that consists of a Page Base Address and an Offset. The size of the Offset field is determined by the physical memory page size configured in CC.MPS.

The Page Base Address and Offset (PBAO) field indicates the 64–bit physical memory page address. The lower bits (*n*:2) of this field indicate the offset within the memory page. If the memory page size is 4KB, then bits 11:02 form the Offset; if the memory page size is 8KB, then bits 12:02 form the Offset, etc (i.e. n = CC.MPS + 11). If this entry is not the first PRP entry in the command or a PRP List pointer in a command, then the Offset portion of this field shall be cleared to 0h.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to issue an Identify command with properly formatted PRP entries to the NVMe Controller.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

Possible Problems: None.

Test 2.50 – Completion Queue Entry (M)

Purpose: To verify that an NVMe Controller can properly post a completion queue entry to the proper Completion Queue.

References:

NVMe Specification 4.6

Resource Requirements:

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

Last Modification: November 30, 2020

Discussion: An entry in the Completion Queue is 16 bytes in size. The NVMe specification describes the layout of the Completion Queue Entry data structure. The contents of Dword 0 are command specific. If a command uses Dword 0, then the definition of this Dword is contained within the associated command definition. If a command does not use Dword 0, then the field is reserved. Dword 1 is reserved. Dword 2 is defined in Figure 26 and Dword 3 is defined in Figure 27 of the NVMe specification. Any additional I/O Command Set defined in the future may use an alternate Completion Queue entry size or format.

The Status Field (SF) of Dword 3 is tested in Test 5.3 – Status Field Definition.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to setup and create an I/O Submission Queue paired with a I/O Completion Queue.
2. Configure the NVMe Host to issue a Write command to the NVMe Controller through the IOSQ created in the previous step.
3. After completion of the Write Command, delete both the I/O Submission Queue and I/O Completion Queue .
4. Configure the NVMe Host to setup and create a new I/O Submission Queue paired with a new I/O Completion Queue.
5. Configure the NVMe Host to issue a Write command to the NVMe Controller through the new IOSQ.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the I/O Completion Queue indicating the status for the command.
2. Verify that the Submission Queue Identifier (SQID) field (bits 31:16) of Dword 2 of each completion queue entry is the SQID of the Submission Queue used to issue the command.
3. Verify that the Phase Tag (P) field (bit 16) of Dword 3 is set to 1 for the first round of completion queue entries arriving at the IOCQ and cleared to 0 for the second round of completion queue entries after the controller had wrapped around the top of the Completion Queue.
4. Verify that the Command Identifier (CID) field (bits 15:00) of Dword 3 for each completion queue entry matches the command ID assigned to the command by the NVMe Host when it was issued to the Submission Queue.

Possible Problems: None.

Test 2.51 – Status Field Definition (M, OF)

Purpose: To verify that an NVMe Controller can properly return the status in the completion queue entry for a command.

References:

NVMe Specification 4.6.1

Resource Requirements:

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

Last Modification: June 24, 2015

Discussion: The Status Field (SF) defines the status for the command indicated in the completion queue entry. If the SCT and SC fields are cleared to 0h, then this indicates a successful command completion, with no fatal or non–fatal error conditions.

The Status Field is bits 31:17 of Dword 3 of all completion queue entries.

The SCT and SC fields are further tested in subsequent tests.

Test Setup: See Appendix A.

Test Procedure:

1. Configure the NVMe Host to issue an Identify command to the NVMe Controller.
2. After the NVMe Controller has processed the command, configure the NVMe Host to reap the completion queue entry for the Identify command from the Admin Completion Queue.

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 Status Code Type (SCT) field of the Status Field is 0h to indicate a Generic Command Status. Verify that the Status Code (SC) field of the Status Field is 0h to indicate successful completion of the command.

Possible Problems: None.

Test 2.52 – Generic Command Status Definition (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly return Status Code values for the Generic Command Status type.

References:

NVMe Specification 4.6.1.2.1

Resource Requirements:

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

Last Modification: April 2, 2019

Discussion: Completion queue entries with a Status Code type of Generic Command Status (0h) indicate a status value associated with the command that is generic across all command types. The Status Code values for the Generic Command Status type are defined in Table 14 and Table 13.

Table 13 – Generic Command Status Values, NVM Command Set

Value	Description
80h	**LBA Out of Range:** The command references an LBA that exceeds the size of the namespace.
81h	**Capacity Exceeded:** Execution of the command has caused the capacity of the namespace to be exceeded. This error occurs when the Namespace Utilization exceeds the Namespace Capacity.
82h	**Namespace Not Ready:** The namespace is not ready to be accessed. The Do Not Retry bit indicates whether re–issuing the command at a later time may succeed.
83h	**Reservation Conflict:** The command was aborted due to a conflict with a reservation held on the accessed namespace.
84h	**Format In Progress.** The namespace is currently being formatted. The Do Not Retry bit shall be cleared to '0' to indicate that the command may succeed if it is resubmitted.
85h – BFh	Reserved

Test Setup: See Appendix A.

Test Procedure:

1. For each of the Status Code values defined in Table 13 and Table 14, configure the NVMe Host to issue a command to the NVMe Controller which will cause the controller to return the Status Code in the completion queue entry for the command.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the SCT field of the Status Field is cleared to 0h. Verify that the SC field of the Status field matches the expected Status Code.

Possible Problems: The NVMe specification does not explicitly state the exact conditions for when an NVMe Controller should use some of the defined Status Codes. Such status codes cannot be tested.

Table 14 – Generic Command Status Values

Value	Description
00h	**Successful Completion:** The command completed successfully.
01h	**Invalid Command Opcode:** The associated command opcode field is not valid.
02h	**Invalid Field in Command:** An invalid field in the command parameters.
03h	**Command ID Conflict:** The command identifier is already in use. Note: It is implementation specific how many commands are searched for a conflict.
04h	**Data Transfer Error:** Transferring the data or metadata associated with a command had an error.
05h	**Commands Aborted due to Power Loss Notification:** Indicates that the command was aborted due to a power loss notification.
06h	**Internal Error:** The command was not completed successfully due to an internal error. Details on the internal device error are returned as an asynchronous event. Refer to section 5.2.
07h	**Command Abort Requested:** The command was aborted due to a Command Abort command being received that specified the Submission Queue Identifier and Command Identifier of this command.
08h	**Command Aborted due to SQ Deletion:** The command was aborted due to a Delete I/O Submission Queue request received for the Submission Queue to which the command was submitted.
09h	**Command Aborted due to Failed Fused Command:** The command was aborted due to the other command in a fused operation failing.
0Ah	**Command Aborted due to Missing Fused Command:** The command was aborted due to the companion fused command not being found as the subsequent Submission Queue entry.
0Bh	**Invalid Namespace or Format:** The namespace or the format of that namespace is invalid.
0Ch	**Command Sequence Error:** The command was aborted due to a protocol violation in a multi–command sequence (e.g. a violation of the Security Send and Security Receive sequencing rules in the TCG Storage Synchronous Interface Communications protocol).
0Dh	**Invalid SGL Last Segment Descriptor:** The command includes an invalid SGL Last Segment. This may occur when the SGL segment pointed to by an SGL Last Se4gment descriptor contains an SGL Segment descriptor or an SGL Last Segment descriptor. This may occur when an SGL Last Segment descriptor contains an invalid length (i.e., a length of zero or one that is not a multiple of 16).
0Eh	**Invalid Number of SGL Descriptors:** The number of SGL descriptors or SGL Last Segment descriptors in a SGL segment is greater than one.
0Fh	**Data SGL Length Invalid:** The length of a Data SGL is too short or too long.
10h	**Metadata SGL Length Invalid:** The length of a Metadata SGL is too short or too long.
11h	**SGL Descriptor Type Invalid:** The type of an SGL Descriptor is a type that is not supported by the controller.
12h	**Invalid Use of Controller Memory Buffer:** The attempted use of the Controller Memory Buffer is not supported by the controller.
13h	**PRP Offset Invalid:** The Offset field for a PRP entry is invalid. This may occur when there is a PRP entry with a non–zero offset after the first entry.
14h	**Atomic Write Unit Exceeded:** The length specified exceeds the atomic write unit size.
15h–7Fh	Reserved
80h–BFh	I/O Command Set Specific
C0h–FFh	Vendor Specific

Test 2.53 – Command Specific Errors Definition (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly return the Status Code values for the Command Specific Status type.

References: 4.6.1.2.2 Figure 33, 5.21.1.9 (Case 7)

Resource Requirements:

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

Last Modification: April 2, 2019

Discussion: Completion queue entries with a Status Code Type of Command Specific Errors (01h) indicate an error that is specific to a particular command opcode. These status values may indicate additional processing is required. The Status Code values for the Command Specific Status Type are shown below.

For each of the Status Code values defined in Table 15 and 17 configure the NVMe Host to issue a command to the NVMe Controller which will cause the controller to return the Status Code in the completion queue entry for the command.

Table 15 – Command Specific Status Values, NVM Command Set

Value	Description	Commands Affected	Test Coverage
80h	Conflicting Attributes	Dataset Management, Read, Write	
81h	Invalid Protection Information	Compare, Read, Write, Write Zeroes	
82h	Attempted Write to Read Only Range	Dataset Management, Write, Write Uncorrectable, Write Zeroes	
83h – BFh	Reserved	N/A	

Table 16 – Command Specific Status Values

Value	Description	Commands Affected	Test Coverage
00h	Completion Queue Invalid	Create I/O Submission Queue	Test 1.4 Case 7
01h	Invalid Queue Identifier	Create I/O Submission Queue, Create I/O Completion Queue, Delete I/O Submission Queue, Delete I/O Completion Queue	Test 1.4 Case 2 and 7
02h	Invalid Queue Size	Create I/O Submission Queue, Create I/O Completion Queue	Test 1.4 Case 4 and 5
03h	Abort Command Limit Exceeded	Abort	Test 5.5 Case 1
04h	Reserved	N/A	N/A
05h	Asynchronous Event Request Limit Exceeded	Asynchronous Event Request	Test 5.5 Case 2
06h	Invalid Firmware Slot	Firmware Commit	Test 5.5 Case 3
07h	Invalid Firmware Image	Firmware Commit	Not Implemented
08h	Invalid Interrupt Vector	Create I/O Completion Queue	Test 1.4 Case 8
09h	Invalid Log Page	Get Log Page	Test 1.3 Case 2 and 3
0Ah	Invalid Format	Format NVM	Test 1.6 Case 5
0Bh	Firmware Activation Requires Conventional Reset	Firmware Commit	Not Implemented
0Ch	Invalid Queue Deletion	Delete I/O Completion Queue	Test 1.4 Case 3

0Dh	Feature Identifier Not Saveable	Set Features	Test 5.5 Case 4
0Eh	Feature Not Changeable	Set Features	Test 5.5 Case 5
0Fh	Feature Not Namespace Specific	Set Features	Test 5.5 Case 6 and 7
10h	Firmware Activation Required NVM Subsystem Reset	Firmware Commit	Not Implemented
11h	Firmware Activation Requires Reset	Firmware Commit	Not Implemented
12h	Firmware Activation Requires Maximum Time Violation	Firmware Commit	Not Implemented
13h	Firmware Activation Prohibited	Firmware Commit	Not Implemented
14h	Overlapping Range	Firmware Commit, Firmware Image Download, Set Features	Test 5.5 Case 8
15h	Namespace Insufficient Capacity	Namespace Management	Test 9.2 Case 3
16h	Namespace Identifier Unavailable	Namespace Management	Test 9.2 Case 1
17h	Reserved	N/A	N/A
18h	Namespace Already Attached	Namespace Attachment	Test 9.3 Case 1
19h	Namespace Is Private	Namespace Attachment	Not Implemented
1Ah	Namespace Not Attached	Namespace Attachment	Test 9.3 Case 2
1Bh	Thin Provisioning Not Supported	Namespace Management	Not Implemented
1Ch	Controller List Invalid	Namespace Attachment	Not Implemented
1Dh	Device Self-test In Progress	Device Self-test	Not Implemented
1Eh	Boot Partition Write Prohibited	Firmware Commit	Not Implemented
1Fh	Invalid Controller Identifier	Virtualization Management	Not Implemented
20h	Invalid Secondary Controller State	Virtualization Management	Not Implemented
21h	Invalid Number of Controller Resources	Virtualization Management	Not Implemented
22h	Invalid Resource Identifier	Virtualization Management	Not Implemented
23h – 6Fh	Reserved	N/A	N/A
70h-7Fh	Directive Specific		
80h – BFh	I/O Command Set Specific		Not Implemented
C0h – FFh	Vendor Specific	N/A	N/A

Test Setup: See Appendix A.

Case 1: **Abort Command Limit Exceeded (M, OF-FYI)**

Test Procedure:

1. Determine the Abort Command Limit Exceeded value, *n*, by examining the Identify Controller Data Structure of the DUT.
2. Configure the NVMe Host to issue *n+1* Abort commands to the NVMe Controller.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the SCT field of the Status Field is set to Abort Command Limit Exceeded, 03h for one command only.

Case 2: **Asynchronous Event Request Limit Exceeded (M, OF-FYI)**

Test Procedure:

1. Determine the Asynchronous Event Request Limit value, *n*, by examining the Identify Controller Data Structure of the DUT.
2. Configure the NVMe Host to issue *n+1* Asynchronous Event Request commands to the NVMe Controller.

Observable Results:
1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the SCT field of the Status Field is set to Asynchronous Event Request Limit Exceeded, 05h for one command only.

Case 3: **Invalid Firmware Slot (M, OF-FYI)**

Test Procedure:
1. Check that OACS Bit 2 = 1. If not, then test is not applicable.
2. Determine the number of firmware slots the DUT supports by examining the Firmware Update field in the Identifier Controller Data Structure.
3. Configure the NVMe Host to issue a Firmware Commit with a Slot ID of one greater than the number of slots supported by the DUT.
    a. If the DUT supports the maximum number of slots, determine if Slot 1 is Read-only. If Slot 1 is read-only, issue a Firmware Commit to Slot 1. If Slot 1 is not read-only, then this test is not applicable.

Observable Results:
1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the SCT field of the Status Field is set to Invalid Firmware Slot, 06h.

Case 4: **Feature Identifier Not Saveable (M, OF-FYI)**

Test Procedure:
1. Configure the NVMe Host to issue a Get Feature command to the NVMe Controller for each supported Feature ID.
2. For each Feature indicated as Not Saveable, issue a Set Feature Command to save that feature.

Observable Results:
1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the SCT field of the Status Field is set to Feature Identifier Not Saveable, 0Dh.

Case 5: **Feature Not Changeable (M, OF-FYI)**

Test Procedure:
5. Configure the NVMe Host to issue a Get Feature command to the NVMe Controller for each supported Feature ID.
6. For each Feature indicated as Not Changeable, issue a Set Feature Command to change the value of that feature to a value different than the current value.

Observable Results:
1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify if the DUT indicates that certain supported features are unchangeable, that the SCT field of the Status Field is set to Feature Not Changeable, 0Eh when the DUT responds to Set Features commands sent for that Feature. If no features are indicated as Not Changeable, then this test case is Not Applicable.

Case 6: **Feature Not Namespace Specific IV=1 (M, OF-FYI)**

Test Procedure:

1. Configure the NVMe Host to issue a Set Feature – Interrupt Vector Configuration command to the NVMe Controller with IV=1 and CD=0.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the SCT field of the Status Field is set to Feature Not Namespace Specific, 0Fh.

Case 7: **Overlapping Range (FYI, OF-FYI)**

Test Procedure:

1. Configure the NVMe Host to issue a Set Feature command to the NVMe Controller with an overlapping LBA range. See Possible Problems description for Case 7 below.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the SCT field of the Status Field is set to Overlapping Range, 14h.

Possible Problems: The NVMe specification does not explicitly state the exact conditions for when an NVMe Controller should use some of the defined Status Codes. Such status codes cannot be tested. Additionally, some of the Status Codes can only be used for optional commands which the NVMe controller may or may not support.

For Case 5, some instances had been observed where if a Feature was not changeable, and the Host tried to change the feature value, but the value offered was the same as the current value, the controller accepted the Set Feature command. If this occurs, the result may be dependent on the order of field checking that the controller uses, which is beyond the scope of the NVMe specification. The test case has been updated to avoid this condition.

For Case 7, requirements on reporting Overlapping range errors were relaxed in NVMe v1.4 relative to earlier versions of the specification. While controllers should report overlapping range errors, it is no longer a ‘shall’ requirements to do so. Therefore this test should be considered FYI, and informative only.

Test 2.54 – Media and Data Integrity Errors Definition (M, OF-FYI)

Purpose: To verify that an NVMe Controller can properly return the status for the command in the completion queue entry.

References:

NVMe Specification 4.6.1.2.3

Resource Requirements:

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

Last Modification: April 2, 2019

Discussion: Completion queue entries with a Status Code Type of Media and Data Integrity Errors (02h) indicate a media specific error that occurred in the NVM or data integrity type errors. The Status Code values for the Media and Data Integrity Errors Status Type are defined in Table 17 and Table 18.

Table 17 – Media and Data Integrity Error Values

Value	Description
00h – 7Fh	Reserved
80h – BFh	I/O Command Set Specific
C0h – FFh	Vendor Specific

Table 18 – Media and Data Integrity Error Values, NVM Command Set

Value	Description
80h	**Write Fault:** The write data could not be committed to the media.
81h	**Unrecovered Read Error:** The read data could not be recovered from the media.
82h	**End–to–end Guard Check Error:** The command was aborted due to an end–to–end guard check failure.
83h	**End–to–end Application Tag Check Error:** The command was aborted due to an end–to–end application tag check failure
84h	**End–to–end Reference Tag Check Error:** The command was aborted due to an end–to–end reference tag check failure.
85h	**Compare Failure:** The command failed due to a miscompare during a Compare command.
86h	**Access Denied:** Access to the namespace and/or LBA range is denied due to lack of access rights. Refer to TCG SIIS.
87h	**Deallocated or Unwritten Logical Block:** The command failed due to an attempt to read from an LBA range containing a deallocated or unwritten logical block.
88h – BFh	Reserved

Test Setup: See Appendix A.

Test Procedure:

1. For each of the Status Code values defined in Table 17 and Table 18, configure the NVMe Host to issue a command to the NVMe Controller which will cause the controller to return the Status Code in the completion queue entry for the command.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the SCT field of the Status Field is set to 2h. Verify that the SC field of the Status field matches the expected Status Code.

Possible Problems: The NVMe specification does not explicitly state the exact conditions for when an NVMe Controller should use some of the defined Status Codes. Such status codes cannot be tested. Additionally, some of the Status Codes can only be used for optional commands which the NVMe controller may or may not support.

Group 4: Controller Architecture

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 2.55 – Controller Level Reset – Conventional Reset (M)

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

References:

NVMe Specification 7.3.2

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 2.56 – Controller Level Reset – Function Level Reset (M)

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

References:

NVMe Specification 7.3.2

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 2.57 – Controller Level Reset – Controller Reset (M, OF)

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

References:

NVMe Specification 7.3.2

Resource Requirements:

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

Last Modification: June 24, 2015

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

A Controller Reset is initiated when the CC.EN controller register field transitions from '1' to '0'. This is performed by the NVMe Host by writing a value of '0' to the CC.EN while the CC.EN field is set to '1'.

Test Setup: See Appendix A.

Test Procedure

1. Configure the NVMe Host to write a value of '0' to the CC.EN controller register field.
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 the Controller 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. The Admin Queue registers (AQA, ASQ, or ACQ) are not reset as part of a controller reset. All other controller registers defined in section 3 of the NVMe Specification and internal controller state are reset.

Possible Problems: None.

Test 2.58 – Controller Level Reset – NVM Subsystem Reset (M)

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

References:

NVMe Specification 7.3.1 and 7.3.2

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 5: Reservations

Overview:

This section describes a method for performing conformance verification for NVMe products implementing NVMe Reservations. These tests are not applicable to devices that do not claim to support reservations.

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 2.59 – Reservation Report Command (M, OF-FYI)

Purpose: To determine if an NVMe Controller properly reports the status of a reservation when processing a Reservation Report command.

References:

NVMe Specification 8.8, 5.14.1.15, 6.13, NVMe v1.3 ECN 003

Resource Requirements:

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

Last Modification: April 8, 2019

Discussion: A host may determine the current reservation status associated with a namespace by executing a Reservation Report command.

The Reservation Report command returns a Reservation Status data structure to memory that describes the registration and reservation status of a namespace.

The size of the Reservation Status data structure is a function of the number of controllers in the NVM Subsystem that are associated with hosts that are registrants of the namespace (i.e., there is a Registered Controller data structure for each such controller).

The command uses Command Dword 10. If the command uses PRPs for the data transfer, then PRP Entry 1 and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the SGL Entry 1 field is used. All other command specific fields are reserved.

Test Setup: See Appendix A.

Case 1: **No Registrants (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
        ii. Configure the NVMe Host to issue a Get Features command with the Reservation Persistence feature to the NVMe Controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Type (RTYPE) field of the Reservation Status data structure is set to 0 to indicate that no reservation is held on the namespace.
3. Verify that the Number of Registered Controllers (REGCTL) field of the Reservation Status data structure is set to 0 to indicate that no hosts are registrants of the namespace.

4. Verify that the Persist Through Power Loss State (PTPLS) field of the Reservation Status data structure is set to the same value of the Persist Through Power Loss (PTPL) field of Command Dword 0 of the completion queue entry for the Get Features command with Reservation Persistence feature.
5. Verify that the controller supports the Host Identifier feature.

Case 2: **Host is a Registrant (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace, and perform a Reservation Acquire to that namespace.
        ii. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
        iii. Configure the NVMe Host to issue an Identify command specifying CNS value 01h to the NVMe Controller in order to receive back the Identify Controller data structure for that controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Generation (GEN) field of the Reservation Status data structure is incremented for each Reservation Register command which the host issues to an NVMe Controller.
3. Verify that the Reservation Type (RTYPE) field of the Reservation Status data structure is set to not 0 to indicate that a reservation is held on the namespace.
4. Verify that the Number of Registered Controllers (REGCTL) field of the Reservation Status data structure is set to the number of controllers for which the host has set its Host Identifier for to indicate the number of controllers associated with the host and that the host is a registrant of the namespace.
5. Verify that the number of Registered Controller data structures returned as part of the Reservation Status data structure is exactly equal to the value stored in the REGCTL field.
6. Verify that the Controller ID (CNTLID) field of the Registered Controller data structures matches the CNTLID field in the Identify Controller data structure for that controller.
7. Verify that the Reservation Status (RCSTS) field of the Registered Controller data structures have bit 0 cleared to not '0' to indicate that the host associated with the controller holds a reservation on the namespace.
8. Verify that the Host Identifier (HOSTID) field of the Registered Controller data structures is set to the same value which the host set for its Host Identifier in the Set Features command.Verify that the Reservation Key (RKEY) field of the Registered Controller data structures is set to the same value which the host set for its reservation key in the Reservation Register command.
9. Verify that the controller supports the Host Identifier feature.

Case 3: **64 Bit Host Identifier (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.

2. Determine if the Controller supports the Host Identifier feature identifier (81h).
3. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to register a 64 bit Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace, and perform a Reservation Acquire to that namespace.
        ii. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller with the Extended Data Structure bit set to 1 in Command Dword 11.
4. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:
1. Verify that the controller aborts the Reservation Report command with the status code of Host Identifier Inconsistent Format.
2. Verify that the controller supports the Host Identifier feature.

Case 4: **128 Bit Host Identifier (FYI, OF-FYI)**

Test Procedure:
1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. Check the CTRATT field to determine if the controller supports 128 bit Host identifiers. If the controller does not support 128 bit host identifiers then this test is not applicable.
3. Determine if the Controller supports the Host Identifier feature identifier (81h).
4. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to register a 128 bit Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace, and perform a Reservation Acquire to that namespace.
        ii. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller with the Extended Data Structure bit set to 0 in Command Dword 11.
5. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:
1. Verify that the controller supports the Host Identifier feature.
2. Verify that the controller aborts the Reservation Report command with the status code of Host Identifier Inconsistent Format.

Case 5: **Dynamic Controller Not Associated with Host (FYI, OF-FYI)**

Test Procedure:
1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:

   a. Check that the controller is a dynamic controller (as defined in the NVMe-oF specification). If the controller is not a dynamic controller then this test is not applicable.
   b. Configure the NVMe Host to issue a Reservation Report command to the dynamic NVMe Controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that the controller sets the Controller ID field to FFFFh.

Possible Problems: None known.

Test 2.60 – Reservation Registration (M, OF-FYI)

Purpose: To determine if an NVMe Controller properly supports registering hosts via the Reservation Register command.

References:

NVMe Specification 8.8 , 5.14.1.15, 6.11

Resource Requirements:

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

Last Modification: April 8, 2019

Discussion: Prior to establishing a reservation on a namespace, a host shall become a registrant of that namespace by registering a reservation key. Registering a reservation key with a namespace creates an association between a host and a namespace. A host need only register on a single controller in order to become a registrant of the namespace on all controllers in the NVM Subsystem that have access to the namespace and are associated with the host.

A host registers a reservation key by executing a Reservation Register command on the namespace with Reservation Register Action (RREGA) field set to 000b (i.e., Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field.

The Reservation Register command uses Command Dword 10 and a Reservation Register data structure in memory. If the command uses PRPs for the data transfer, then the PRP Entry 1 and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the SGL Entry 1 field is used. All other command specific fields are reserved.

Test Setup: See Appendix A.

Case 1: **Basic Operation (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each shared namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        ii. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that the host was successfully registered to the namespace.

Case 2: **Re-registration (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each shared namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        ii. Configure the NVMe Host to issue an additional Reservation Register command with Register Reservation Key action and the same reservation key to the NVMe Controller.
        iii. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
        iv. Configure the NVMe Host to issue an additional Reservation Register command with Register Reservation Key action and a different reservation key to the NVMe Controller.
        v. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that the host was successfully registered to the namespace.
3. Verify that the completion queue entry for the final Reservation Register command (with the different reservation key) indicates status Reservation Conflict.
4. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the final Reservation Report command indicates that the reservation key for the host was not changed.

Case 3: **Replace Registration Key (M, OF-FYI)**

A host that is a registrant of a namespace may replace its existing reservation key by executing a Reservation Register command on the namespace with the RREGA field set to 010b (i.e., Replace Reservation Key), supplying the current reservation key in the Current Reservation Key (CRKEY) field, and the new reservation key in the NRKEY field. If the contents of the CRKEY field do no match the key currently associated with the host, then the Reservation Register command shall be aborted with status of Reservation Conflict. A host may replace its reservation key without regard to its registration status or current reservation key value by setting the Ignore Existing Key (IEKEY) bit to '1' in the Reservation Register command. Setting the IEKEY bit to '1' causes the Reservation Register command to succeed regardless of the value of the CRKEY field (i.e., the current reservation key is not checked).

Replacing a reservation key has no effect on any reservation that may be held on the namespace.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each shared namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying

a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.

ii. Configure the NVMe Host to issue a Reservation Register command with the RREGA field set to 010b (i.e. Replace Reservation Key), supplying the current reservation key in the Current Reservation Key (CRKEY) field, and a new reservation key in the NRKEY field to the NVMe Controller for the namespace in order to associate a new reservation key with the registrant of the namespace.

iii. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.

iv. Configure the NVMe Host to issue a Reservation Register command with the Replace Reservation Key action, supplying any key value that is not the current reservation key in the CRKEY field, and a new reservation key in the NRKEY field to the NVMe Controller. Also, set the RType to 2 using the Reservation Acquire Command.

v. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.

vi. Configure the NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field to the NVMe Controller in order for the host to acquire a reservation on the namespace.

vii. Configure the NVMe Host to issue a Reservation Register command with the Register Reservation Key action, supplying a new reservation key in the New Reservation Key (NRKEY) field, and setting the IEKEY bit to '1' to the NVMe Controller for the namespace in order to make the host a registrant of that namespace with a different reservation key.

viii. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.

3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command after the second Reservation Registration command indicates that the host successfully changed its reservation key.
3. Verify that the completion queue entry for the third Reservation Register command (with the invalid current reservation key) indicates status Reservation Conflict.
4. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command after the third Reservation Register command indicates that the reservation key associated with the host did not change.
5. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command after the fourth Reservation Register command (with the IEKEY set) indicates that the host successfully changed its reservation key and that it still holds the reservation it previously acquired.

Case 4: **Multiple Hosts (FYI) Dual Port Devices Only**

There are no restrictions on the reservation key value used by hosts with different Host Identifiers. For example, multiple hosts may all register the same reservation key value.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.

3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying the same reservation key that NVMe Host 1 used in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
6. Configure NVMe Host 1 to issue a Reservation Report command to NVMe Controller 1.
7. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that both hosts were successfully registered to the namespace.

Case 5: **Reservation Persistence (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem, configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
3. For each shared namespace attached to the NVMe Controller:
    a. Configure the NVMe Host to issue a Set Features command with the Reservation Persistence feature in order to set PTPL to 0 for that Namespace. If this feature is not changeable then this test is not applicable.
    b. Configure the NVMe Host to issue a Get Features command for the Reservation Persistence feature. Verify that the previously supplied value was returned.
    c. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace, and set CPTPL to 00b (No change to PTPL state).
    d. Configure the NVMe Host to issue a Get Features command for the Reservation Persistence feature. Verify that the previously supplied value for PTPL (0) was returned.
    e. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace, and set CPTPL to 11b (Set PTPL state to ‘1’).
    f. Configure the NVMe Host to issue a Get Features command for the Reservation Persistence feature. Verify that the value returned for PTPL is 1.
    g. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace, and set CPTPL to 10b (Set PTPL state to ‘0’).
    h. Configure the NVMe Host to issue a Get Features command for the Reservation Persistence feature. Verify that the value returned for PTPL is 0.
    i. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
4. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that in steps 3b, 3e, and 3i, the DUT returned a value of 0 for the PTPL value in the Get Feature response.
2. Verify that in step 3g, the DUT returned a value of 1 for the PTPL value in the Get Feature response.

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 2.61 – Unregistering (M, OF-FYI)

Purpose: To determine if an NVMe Controller properly supports reservations.

References:

NVMe Specification 8.8 , 5.14.1.15, 6.11

Resource Requirements:

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

Last Modification: April 8, 2019

Discussion: A host that is a registrant of a namespace may unregister with the namespace by executing a Reservation Register command on the namespace with the Reservation Register Action (RREGA) field set to 0001b (i.e., Unregister Reservation Key) and supplying its current reservation key in the CRKEY field. If the contents of the CRKEY field do not match the key currently associated with the host or if the host is not a registrant, then the command shall be aborted with a status of Reservation Conflict. A host may unregister without regard to its current reservation key value by setting the IEKEY bit to '1' in the Reservation Register command.

Successful completion of an unregister operation causes the host to no longer be a registrant of that namespace.

Unregistering due to preemption or a registration clear is verified in subsequent tests.

Test Setup: See Appendix A.

Case 1: **Unregistering with Reservation Register Command (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        ii. Configure the NVMe Host to issue a Reservation Register command with the Unregister Reservation Key action and supplying its current reservation key in the CRKEY field to the NVMe Controller for the namespace in order to unregister the host as a registrant of that namespace.
        iii. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that the host was successfully unregistered from the namespace.

Case 2: **Unregistering due to Preemption (FYI, OF-FYI) Dual Port Devices Only**

If a preemption occurs and there is no reservation held on the namespace, then execution of the Reservation Acquire command with Preempt action causes registrants whose reservation key match the value of the PRKEY field to be unregistered.

See the Preempting a Reservation test for more information on reservation preemption.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.
3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
6. Configure NVMe Host 2 to issue a Reservation Acquire command, setting the RACQA field to 001b (Preempt), and supplying the current reservation key associated with the NVMe Host 2 in the CRKEY field and setting the Preempt Reservation Key (PRKEY) field to the current reservation key associated with NVMe Host 1 to NVMe Controller 2 in order to preempt the NVMe Host 1's reservation.
7. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.
8. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that NVMe Host 2 was not unregistered from the namespace and NVMe Host 1 was successfully unregistered from the namespace.

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 2.62 – Acquiring a Reservation (M, OF-FYI)

Purpose: To determine if an NVMe Controller properly allows acquisition of reservations via the Reservation Register command.

References:

NVMe Specification 8.8, 5.14.1.15, 6.10

Resource Requirements:

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

Last Modification: April 8, 2019

Discussion: In order for a host to obtain a reservation on a namespace, it shall be a registrant of that namespace. A registrant obtains a reservation by executing a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), and supplying the current reservation key associated with the host to the Current Reservation Key (CRKEY) field.

The Reservation Acquire command uses Command Dword 10 and a Reservation Acquire data structure in memory. If the command uses PRPs for the data transfer, then the PRP Entry 1 and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the SGL Entry 1 field is used. All other command specific fields are used.

Test Setup: See Appendix A.

Case 1: **Basic Operation (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        ii. Configure the NVMe Host to issue an Identify command specifying CNS value 00h to the NVMe Controller in order to receive back the Identify Namespace data structure for the namespace.
        iii. For each reservation type supported by the namespace based on the RESCAP field of the Identify Namespace data structure for the namespace:
            1. Configure the NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to the reservation type to the NVMe Controller in order for the host to acquire a reservation on the namespace with the reservation type.
            2. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
            3. Configure the NVMe Host to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to the type of the reservation being released, and supplying the current reservation key associated with the host in the Current

Reservation Key (CRKEY) field to the NVMe Controller in order to release the reservation held by the host.

3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that the host successfully acquired the reservation with the associated reservation type.

Case 2: **Error Conditions (M, OF-FYI)**

If the CRKEY value does not match that used by the registrant to register with the namespace or the host is not a registrant, the command shall be aborted with status Registration Conflict. A Host may acquire a reservation without regard to its current reservation key value by setting the Ignore Existing Key (IEKEY) bit to '1' in the command.

If a reservation holder attempts to obtain a reservation of a different type on a namespace for which it is already the reservation holder, then the command shall be aborted with status Reservation Conflict. It is not an error for a reservation holder to attempt to obtain a reservation of the same type on a namespace for which it is already the reservation holder.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. RES_ACQ1: Configure the NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying a random reservation key in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to the NVMe Controller.
        ii. RES_REP1: Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
        iii. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        iv. RES_ACQ2: Configure the NVMe Host to issue a Reservation Acquire command, setting the Acquire action, supplying any reservation key other than the reservation key currently associated with the host in the CRKEY field, and setting the RTYPE field to a reservation type supported by the namespace to the NVMe Controller.
        v. RES_REP2: Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
        vi. RES_ACQ3: Configure the NVMe Host to issue a Reservation Acquire command, setting the Acquire action, setting the IEKEY bit to '1', and setting the RTYPE field to a reservation type supported by the namespace to the NVMe Controller in order for the host to acquire a reservation on the namespace.
        vii. RES_REP3: Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
        viii. RES_ACQ4: Configure the NVMe Host to issue a Reservation Acquire command, setting the Acquire action, supplying the reservation key currently associated with the host in the CRKEY field, and setting the RTYPE field to a different reservation type supported by the

namespace than the one used for the current reservation held by the host to the NVMe Controller.

ix. RES_REP4: Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.

x. RES_ACQ5: Configure the NVMe Host to issue a Reservation Acquire command, setting the Acquire action, supplying the reservation key currently associated with the host in the CRKEY field, and setting the RTYPE field to the same reservation type used for the current reservation held by the host to the NVMe Controller.

xi. RES_REP5: Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.

3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the completion queue entry for RES_ACQ1 indicates status Reservation Conflict.
3. Verify that the Reservation Status data structure returned by the NVMe Controller after completing RES_REP1 indicates that the host does not hold any reservations.
4. Verify that the completion queue entry for RES_ACQ2 indicates status Reservation Conflict.
5. Verify that the Reservation Status data structure returned by the NVMe Controller after completing RES_REP2 indicates that the host does not hold any reservations.
6. Verify that the completion queue entry for RES_ACQ3 indicates status "Invalid Field in Command".
7. Verify that the Reservation Status data structure returned by the NVMe Controller after completing RES_REP3 indicates that the host does not hold any reservations.
8. Verify that the completion queue entry for RES_ACQ4 indicates status Reservation Conflict.
9. Verify that the Reservation Status data structure returned by the NVMe Controller after completing RES_REP4 indicates that the host still holds its reservation with the namespace with the same reservation type. Verify that the Reservation Status data structure returned by the NVMe Controller after completing RES_REP5 indicates that the host still holds its reservation with the namespace with the same reservation type.

Case 3: **Multiple Hosts (FYI, OF-FYI) Dual Port Devices Only**

Only one reservation is allowed at a time on a namespace. If a registrant attempts to obtain a reservation on a namespace that already has a reservation holder, then the command is aborted with status Reservation Conflict.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable. If the DUT is a single port device then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.
3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
6. Configure NVMe Host 1 to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the reservation key currently associated with NVMe Host 1 in

the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to NVMe Controller 1.

7. Configure NVMe Host 2 to issue a Reservation Acquire command with the Acquire action, supplying the reservation key currently associated with NVMe Host 2 in the CRKEY field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to NVMe Controller 2.
8. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.
9. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the completion queue entry for the Reservation Acquire command sent by NVMe Host 2 indicates status Reservation Conflict.
3. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that NVMe Host 1 holds a reservation on the namespace and that NVMe Host 2 does not.

Possible Problems: None.

Test 2.63 – Releasing a Reservation (M, OF-FYI)

Purpose: To determine if an NVMe Controller properly releases reservations.

References:

NVMe Specification 8.8 , 5.14.1.15, 6.12

Resource Requirements:

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

Last Modification: April 8, 2019

Discussion: A host releases a reservation by executing a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to the type of the reservation being released, and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field.

The Reservation Release command uses Command Dword 10 and a Reservation Release data structure in memory. If the command uses PRPs for the data transfer, then the PRP Entry 1 and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the SGL Entry 1 field is used. All other command specific fields are reserved.

Reservation release due to preemption or a registration clear is verified in subsequent tests.

Test Setup: See Appendix A.

Case 1: **Release with Reservation Release Command (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        ii. Configure the NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to the NVMe Controller in order for the host to acquire a reservation on the namespace with the reservation type.
        iii. Configure the NVMe Host to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to the type of the reservation being released, and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field to the NVMe Controller in order to release the reservation held by the host.
        iv. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that the host successfully released the reservation.

Case 2: **Reservation Release Command Error Conditions (M, OF-FYI)**

If the CRKEY value does not match that used by the registrant to register with the namespace, the command shall be aborted with status Registration Conflict. A host may release a reservation without regard to its current reservation key value by setting the Ignore Existing Key (IEKEY) bit to '1' in the command. If the RTYPE field does not match the type of the current reservation, then the command shall be completed with status Invalid Field in Command.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        ii. Configure the NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to the NVMe Controller in order for the host to acquire a reservation on the namespace with the reservation type.
        iii. RES_REL1: Configure the NVMe Host to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to the type of the reservation being released, and supplying any reservation key other than the current reservation key associated with the host in the Current Reservation Key (CRKEY) field to the NVMe Controller.
        iv. RES_REP1: Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
        v. RES_REL2: Configure the NVMe Host to issue a Reservation Release command with the Release action, setting the RTYPE field to any reservation type other than the type of the reservation being released, and supplying the current reservation key associated with the host in the CRKEY field to the NVMe Controller.
        vi. RES_REP2: Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
        vii. RES_REL3: Configure the NVMe Host to issue a Reservation Release command with the Release action, setting the Reservation Type (RTYPE) field to the type of the reservation being released, supplying the correct reservation key that is associated with the host in the CRKEY field, and setting the IEKEY bit to '1' to the NVMe Controller in order to release the reservation held by the host.
        viii. RES_REP3: Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
        ix. RES_REL4: Configure the NVMe Host to issue a Reservation Release command with the Release action, setting the Reservation Type (RTYPE) field to the type of the reservation being released, supplying the correct reservation key that is associated with the host in the CRKEY field, and setting the IEKEY bit to '0' to the NVMe Controller in order to release the reservation held by the host.

x. RES_REP4: Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.

xi.

3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the completion queue entry for RES_REL1 indicates status Reservation Conflict.
3. Verify that the Reservation Status data structure returned by the NVMe Controller after completing RES_REP1 indicates that the host still holds a reservation with the namespace.
4. Verify that the completion queue entry for RES_REL2 indicates status Invalid Field in Command.
5. Verify that the Reservation Status data structure returned by the NVMe Controller after completing RES_REP2 indicates that the host still holds a reservation with the namespace.
6. Verify that the Reservation Status data structure returned by the NVMe Controller after completing RES_REP3 indicates that the host did not release the reservation, and the Reservation Release command completed with status Invalid Field in Command.
7. Verify that the Reservation Status data structure returned by the NVMe Controller after completing RES_REP4 indicates that the host successfully released the reservation.

Case 3: **Multiple Hosts (FYI, OF-FYI) Dual Port Devices Only**

An attempt by a registrant to release a reservation using the Reservation Release command in the absence of a reservation held on the namespace or when the host is not the reservation holder shall cause the command to complete successfully, but shall have no effect on the controller or namespace.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable. If the DUT is a single port device then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.
3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
6. Configure NVMe Host 1 to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to any reservation type, and supplying the current reservation key associated with NVMe Host 1 in the Current Reservation Key (CRKEY) field to NVMe Controller 1.
7. Configure NVMe Host 1 to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the reservation key currently associated with NVMe Host 1 in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to NVMe Controller 1.
8. Configure NVMe Host 2 to issue a Reservation Release command with the Release action, setting the RTYPE field to the type of the reservation which NVMe Host 1 holds, and supplying the current reservation key associated with NVMe Host 1 in the CRKEY field to NVMe Controller 2.
9. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.

10. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:
1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the completion queue entry for the Reservation Release command sent by NVMe Host 1 indicates status Success.
3. Verify that the completion queue entry for the Reservation Release command sent by NVMe Host 2 indicates status Success.
4. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that NVMe Host 1 holds a reservation on the namespace and that NVMe Host 2 does not.

Case 4: **Release Due to Unregister (M, OF-FYI)**

If a host is the last remaining reservation holder (i.e. the Reservation Type is Write Exclusive - All Registrants or Exclusive Access - All Registrants) or is the only reservation holder, then the reservation is released when the host unregisters.

Test Procedure:
1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        ii. Configure the NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to 01h (Write Exclusive Access) to the NVMe Controller in order for the host to acquire a reservation on the namespace.
        iii. Configure the NVMe Host to issue a Reservation Register command with the Unregister Reservation Key action and supplying its current reservation key in the CRKEY field to the NVMe Controller for the namespace in order to unregister the host as a registrant of that namespace.
        iv. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:
1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that the host was successfully unregistered from the namespace and that the reservation held by the host on the namespace was released.

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 2.64 – Preempting a Reservation (FYI, OF-FYI)

Purpose: To determine if an NVMe Controller properly preempts reservations.

References:

NVMe Specification 8.8, 5.14.1.15, 6.10

Resource Requirements:

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

Last Modification: April 8, 2019

Discussion: A host that is a registrant may preempt a reservation and/or registration by executing a Reservation Acquire command, setting the Reservation Acquire (RACQA) field to 001b (Preempt), and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field. The preempt actions that occur are dependent on the type of reservation held on the namespace, if any, and the value of the Preempt Reservation Key (PKEY) field in the command.

If the CRKEY value does not match, then the command is aborted with status Reservation Conflict.

The case of preemption when no reservation is held is covered in the Unregistering test.

Test Setup: See Appendix A.

Case 1: **Write Exclusive - All Registrants or Exclusive Access - All Registrants (FYI, OF-FYI) Dual Port Devices Only**

If the existing reservation type is Write Exclusive - All Registrants or Exclusive Access - All Registrants, then the actions performed by the command depend on the value of the PRKEY field as follows:

1. If the PRKEY field value is zero, then the following occurs as an atomic operation:
    a. All registrants other than the host that issued the command are unregistered,
    b. The reservation is released, and
    c. A new reservation is created for the host of the type specified by the Reservation Type (RTYPE) field in the command.
2. If the PRKEY value is non-zero, then each registrant whose reservation key matches the value of the PRKEY field are unregistered.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.
3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
6. Configure NVMe Host 1 to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the reservation key currently associated with NVMe Host 1 in

the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to Write Exclusive - All Registrants or Exclusive Access - All Registrants to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation on the namespace.
7. Configure NVMe Host 2 to issue a Reservation Acquire command, setting the RACQA field to 001b (Preempt), and supplying the current reservation key associated with the NVMe Host 2 in the CRKEY field and setting the Preempt Reservation Key (PRKEY) field to 0 to NVMe Controller 2 in order to preempt NVMe Host 1's reservation.
8. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.
9. Configure NVMe Host 2 to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to the type of the reservation being released, and supplying the current reservation key associated with NVMe Host 2 in the Current Reservation Key (CRKEY) field to NVMe Controller 2 in order to release the reservation held by NVMe Host 2.
10. Configure NVMe Host 1 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
11. Configure NVMe Host 1 to issue a Reservation Acquire command with the Acquire action, supplying the reservation key currently associated with NVMe Host 1 in the CRKEY field, and setting the RTYPE field to Write Exclusive - All Registrants or Exclusive Access - All Registrants to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation on the namespace.
12. Configure NVMe Host 2 to issue a Reservation Acquire command with the Preempt action, and supplying the current reservation key associated with the NVMe Host 2 in the CRKEY field and setting the PRKEY field to the current reservation key associated with NVMe Host 1 to NVMe Controller 2 in order to preempt NVMe Host 1's reservation.
13. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.
14. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:
1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the first Reservation Report command indicates that NVMe Host 2 was not unregistered from the namespace, NVMe Host 1 no longer holds a reservation and was successfully unregistered from the namespace, and that NVMe Host 2 holds a reservation of the type specified in the Reservation Acquire command with Preempt action.
3. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the second Reservation Report command indicates that NVMe Host 2 was not unregistered from the namespace and NVMe Host 1 was successfully unregistered from the namespace.

Case 2: **Other Registration Types (FYI, OF-FYI) Dual Port Devices Only**

If the existing reservation type is not Write Exclusive - All Registrants and not Exclusive Access - All Registrants (i.e. is Write Exclusive, Exclusive Access, Write Exclusive - Registrants Only, or Exclusive Access - Registrants Only), then the actions performed by the command depend on the value of the PRKEY as follows:
1. If the PRKEY field value matches the reservation key of the current reservation key of the current reservation holder, then the following occur as an atomic operation:
    a. The reservation holder is unregistered,
    b. The reservation is released, and
    c. A new reservation is created of the type specified by the Reservation Type (RTYPE) field in the command for the host as the reservation key holder.
2. If the PRKEY value does not match that of the current reservation holder and is not equal to zero, then each registrant whose reservation key matches the value of the value of the PRKEY field are unregistered.
3. If the PRKEY value does not match that of the current reservation holder and is equal to zero, then the command shall be aborted with status Invalid Field in Command.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable. If the DUT is a single port device then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.
3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
6. Configure NVMe Host 1 to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the reservation key currently associated with NVMe Host 1 in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to Write Exclusive, Exclusive Access, Write Exclusive - Registrants Only, or Exclusive Access - Registrants Only to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation on the namespace.
7. Configure NVMe Host 2 to issue a Reservation Acquire command, setting the RACQA field to 001b (Preempt), and supplying the current reservation key associated with the NVMe Host 2 in the CRKEY field and setting the Preempt Reservation Key (PRKEY) field to the reservation key associated with NVMe Host 1 to NVMe Controller 2 in order to preempt NVMe Host 1's reservation.
8. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.
9. Configure NVMe Host 2 to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to the type of the reservation being released, and supplying the current reservation key associated with NVMe Host 2 in the Current Reservation Key (CRKEY) field to NVMe Controller 2 in order to release the reservation held by NVMe Host 2.
10. Configure NVMe Host 1 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
11. Configure NVMe Host 1 to issue a Reservation Acquire command with the Acquire action, supplying the reservation key currently associated with NVMe Host 1 in the CRKEY field, and setting the RTYPE field to Write Exclusive, Exclusive Access, Write Exclusive - Registrants Only, or Exclusive Access - Registrants Only to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation on the namespace.
12. Configure NVMe Host 2 to issue a Reservation Acquire command with the Preempt action, and supplying the current reservation key associated with the NVMe Host 2 in the CRKEY field and setting the PRKEY field to a reservation key other than NVMe Host 1 or NVMe Host 2's reservation keys to NVMe Controller 2.
13. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.
14. Configure NVMe Host 1 to issue a Reservation Acquire command with the Acquire action, supplying the reservation key currently associated with NVMe Host 1 in the CRKEY field, and setting the RTYPE field to Write Exclusive, Exclusive Access, Write Exclusive - Registrants Only, or Exclusive Access - Registrants Only to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation on the namespace.
15. Configure NVMe Host 2 to issue a Reservation Acquire command with the Preempt action, and supplying the current reservation key associated with the NVMe Host 2 in the CRKEY field and setting the PRKEY field to 0 to NVMe Controller 2 in order to preempt NVMe Host 1's reservation.
16. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.
17. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the first Reservation Report command indicates that NVMe Host 2 was not unregistered from the namespace, NVMe Host 1 no longer holds a reservation and was successfully unregistered from the namespace, and that NVMe Host 2 holds a reservation of the type specified in the Reservation Acquire command with Preempt action.
3. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the second Reservation Report command indicates that both hosts are still registered and NVMe Host 1 still holds a reservation on the namespace.
4. Verify that the completion queue entry for the third Reservation Acquire command with Preempt action indicates status Invalid Field in Command.
5. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the third Reservation Report command indicates that both hosts are still registered and NVMe Host 1 still holds a reservation on the namespace.

Case 3: **Self-preemption (FYI, OF-FYI) Dual Port Devices Only**

A reservation holder may preempt itself using the above mechanism. When a host preempts itself, the following occurs as an atomic operation:

1. Registration of the host is maintained,
2. The reservation is released, and
3. A new reservation is created for the host of the type specified by the RTYPE field.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable. If the DUT is a single port device then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        ii. Configure the NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to the NVMe Controller of either WRITE EXCLUSIVE ALL REGISTRANTS or EXCLUSIVE ACCESS ALL REGISTRANTS.
        iii. Configure the NVMe Host to issue a Reservation Acquire command, setting the RACQA field to 001b (Preempt), and supplying the current reservation key associated with the host in the CRKEY field and setting the Preempt Reservation Key (PRKEY) field to 0 to the NVMe Controller in order to preempt the host's reservation.
        iv. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that the host was not unregistered from the namespace and that the host holds a reservation of the type specified in the Reservation Acquire command with Preempt action.

Case 4: **Preempt and Abort (FYI, OF-FYI) Dual Port Devices Only**

A host may abort commands as a side effect of preempting a reservation by executing a Reservation Acquire command and setting the RACQA field to 010b (Preempt an Abort). The behavior of such a command is exactly the same as that described above with the RACQA field set to 001b (Preempt), except that commands that target the namespace are aborted by controllers associated with hosts whose reservation or registration is preempted. As with the Abort Admin command, abort as a side effect of preempting a reservation is best effort; the commands to abort may have already completed, currently be in execution, or may be deeply queued.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable. If the DUT is a single port device then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.
3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
6. Configure NVMe Host 1 to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the reservation key currently associated with NVMe Host 1 in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation on the namespace.
7. Configure NVMe Host 1 to issue 10 NVMe Read commands to NVMe Controller 1.
8. Configure NVMe Host 2 to issue a Reservation Acquire command, setting the RACQA field to 010b (Preempt and Abort), and supplying the current reservation key associated with the NVMe Host 2 in the CRKEY field and setting the Preempt Reservation Key (PRKEY) field to the reservation key associated with NVMe Host 1 to NVMe Controller 2 in order to preempt NVMe Host 1's reservation.
9. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.
10. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Determine the status of each of the NVMe commands issued by NVMe Host 1.
3. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that the proper actions were taken according to the reservation type acquired by NVMe Host 1.

Case 5: **Preempt Attempt when CRKEY does not Match (FYI, OF-FYI) Dual Port Devices Only**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.

3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
6. Configure NVMe Host 1 to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the reservation key currently associated with NVMe Host 1 in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to Write Exclusive - All Registrants or Exclusive Access - All Registrants to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation on the namespace.
7. Configure NVMe Host 2 to issue a Reservation Acquire command, setting the RACQA field to 001b (Preempt), and supplying the current reservation key associated with the NVMe Host 2 in the CRKEY field and setting the Preempt Reservation Key (PRKEY) field to 0 to NVMe Controller 2 in order to preempt NVMe Host 1's reservation.
8. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.
9. Configure NVMe Host 2 to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to the type of the reservation being released, and supplying the current reservation key associated with NVMe Host 2 in the Current Reservation Key (CRKEY) field to NVMe Controller 2 in order to release the reservation held by NVMe Host 2.
10. Configure NVMe Host 1 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
11. Configure NVMe Host 1 to issue a Reservation Acquire command with the Acquire action, supplying the reservation key currently associated with NVMe Host 1 in the CRKEY field, and setting the RTYPE field to Write Exclusive - All Registrants or Exclusive Access - All Registrants to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation on the namespace.
12. Configure NVMe Host 2 to issue a Reservation Acquire command with the Preempt action, and supplying an incorrect key in the CRKEY field and setting the PRKEY field to the current reservation key associated with NVMe Host 1 to NVMe Controller 2 in order to preempt NVMe Host 1's reservation.
13. Configure NVMe Host 2 to issue a Reservation Report command to NVMe Controller 2.
14. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that the Reservation Acquire command with Preempt Action and the incorrect CRKEY is aborted with status Reservation Conflict.

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 2.65 – Clearing a Reservation (M, OF-FYI)

Purpose: To determine if an NVMe Controller properly supports clearing reservations.

References:

NVMe Specification 8.8, 5.14.1.15, 6.12

Resource Requirements:

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

Last Modification: November 26, 2018

Discussion: A host that is a registrant may clear a reservation (i.e. force the release of a reservation held on the namespace and unregister all registrants) by executing a Reservation Release command, setting the Reservation Release Action (RRELA) field to 001b (i.e. Clear), and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field.

When a reservation is cleared, the following occur as an atomic operation:

1. on held on the namespace is released, and
2. All registrants are unregistered from the namespace.

Test Setup: See Appendix A.

Case 1: **Basic Operation with Reservation Release Command (M, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace. Configure the NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to the NVMe Controller in order for the host to acquire a reservation on the namespace with the reservation type.
        ii. Configure the NVMe Host to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 001b (i.e. Clear) and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field to the NVMe Controller in order to clear the reservation.
        iii. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.

2. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the Reservation Report command indicates that the host no longer holds a reservation on the namespace and was successfully unregistered from the namespace.

Case 2: **Error Conditions (M, OF-FYI)**

If the CRKEY value does not match that used by the host to register with the namespace, then the command shall be aborted with status Reservation Conflict. A host may clear a reservation without regard to its current reservation key value by setting the Ignore Existing Key (IEKEY) bit to '1' in the command.

If the host is not a registrant, then the command shall be aborted with a status of Reservation Conflict.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each active namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 001b (i.e. Clear) and setting the Current Reservation Key (CRKEY) field to a random reservation key value to the NVMe Controller.
        ii. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        iii. Configure the NVMe Host to issue a Reservation Release command with the Clear action and supplying any reservation key other than the current reservation key associated with the host in the CRKEY field to the NVMe Controller.
        iv. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
        v. Configure the NVMe Host to issue a Reservation Release command with the Clear action, supplying any reservation key other than the current reservation key associated with the host in the CRKEY field, and setting the IEKEY bit to '1' to the NVMe Controller in order to clear the reservation.
        vi. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
3. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the completion queue entry for the first Reservation Release command with Clear action indicates status Reservation Conflict.
3. Verify that the completion queue entry for the second Reservation Release command with Clear action indicates status Reservation Conflict.
4. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the first Reservation Report command indicates that the host is still a registrant of the namespace.
5. Verify that the Reservation Status data structure returned by the NVMe Controller after completing the second Reservation Report command indicates that the host was successfully unregistered from the namespace.

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 2.66 – Command Behavior with Different Reservation Types (M, OF-FYI)

Purpose: To determine if an NVMe Controller exhibits proper command behavior in the presence of different reservation types.

References:

NVMe Specification 8.8, 5.14.1.15

Resource Requirements:

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

Last Modification: April 8, 2019

Discussion: NVMe supports six types of reservations:

- Write Exclusive
- Exclusive Access
- Write Exclusive - Registrants Only
- Exclusive Access - Registrants Only
- Write Exclusive - All Registrants
- Exclusive Access - All Registrants

The differences between these reservation types are: the type of access that is excluded (i.e., writes or all accesses), whether registrants have the same access rights as the reservation holder, and whether registrants are also considered to be reservation holders.

For the purposes of reservation types, the following commands are considered to be in the NVM Read Command Group:

- Read
- Compare
- Security Receive

And the following commands are considered to be in the NVM Write Command Group:

- Write
- Write Uncorrectable
- Dataset Management
- Flush
- Format NVM
- Namespace Attachment
- Namespace Management
- Security Send

Additionally, certain reservation commands have specific behavior and all other commands shall be allowed regardless of reservation status or type. The behavior of vendor specific commands is vendor specific.

Test Setup: See Appendix A.

Case 1: **Write Exclusive (M, OF-FYI) Dual Port Devices Only**

A Write Exclusive reservation disallows commands from the NVM Write Command Group by any host other than the registration holder. Any commands from the NVM Read Command Group are still allowed by any host.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable. If the DUT is a single port device then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.
3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 1 to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the reservation key currently associated with NVMe Host 1 in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to Write Exclusive to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation of type Write Exclusive.
6. Configure each host to issue each of the commands in the NVM Read Command Group above to their respective controllers.
7. Configure each host to issue each of the commands in the NVM Write Command Group above to their respective controllers.
8. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
9. Configure NVMe Host 2 to issue each of the commands in the NVM Read Command Group above to NVMe Controller 2.
10. Configure NVMe Host 2 to issue each of the commands in the NVM Write Command Group above to NVMe Controller 2.
11. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the completion queue entries for all commands sent by NVMe Host 1 indicate status Successful Completion.
3. Verify that the completion queue entries for all commands from the NVM Read Command Group sent by NVMe Host 2 indicate status Successful Completion.
4. Verify that the completion queue entries for all commands from the NVM Write Command Group sent by NVMe Host 2 indicate status Reservation Conflict.

Case 2: **Exclusive Access (M , OF-FYI) Dual Port Devices Only**

An Exclusive Access reservation disallows commands from both the NVM Write Command Group and the NVM Read Command Group by any host other than the registration holder.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable. If the DUT is a single port device then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.
3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.

5. Configure NVMe Host 1 to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the reservation key currently associated with NVMe Host 1 in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to Exclusive Access to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation of type Exclusive Access.
6. Configure each host to issue each of the commands in the NVM Read Command Group above to their respective controllers.
7. Configure each host to issue each of the commands in the NVM Write Command Group above to their respective controllers.
8. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
9. Configure NVMe Host 2 to issue each of the commands in the NVM Read Command Group above to NVMe Controller 2.
10. Configure NVMe Host 2 to issue each of the commands in the NVM Write Command Group above to NVMe Controller 2.
11. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the completion queue entries for all commands sent by NVMe Host 1 indicate status Successful Completion.
3. Verify that the completion queue entries for all commands sent by NVMe Host 2 indicate status Reservation Conflict.

Case 3: **Write Exclusive - Registrants Only or Write Exclusive - All Registrants (M, OF-FYI) Dual Port Devices Only**

A Write Exclusive - Registrants Only or a Write Exclusive - All Registrants reservation disallows commands from the NVM Write Command Group by any non-registrant of the namespace. Any commands from the NVM Read Command Group are still allowed by any host and registrants of the namespace are still allowed to issue commands from the NVM Write Command Group.

The difference between the Write Exclusive - Registrants Only and Write Exclusive - All Registrants reservation types is that all registrants are also considered reservation holders with the Write Exclusive - All Registrants type.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable. If the DUT is a single port device then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.
3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 1 to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the reservation key currently associated with NVMe Host 1 in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to Write Exclusive - Registrants Only or Write Exclusive - All Registrants to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation of the specified type.

6. Configure each host to issue each of the commands in the NVM Read Command Group above to their respective controllers.
7. Configure each host to issue each of the commands in the NVM Write Command Group above to their respective controllers.
8. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
9. Configure NVMe Host 2 to issue each of the commands in the NVM Read Command Group above to NVMe Controller 2.
10. Configure NVMe Host 2 to issue each of the commands in the NVM Write Command Group above to NVMe Controller 2.
11. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the completion queue entries for all commands sent by NVMe Host 1 indicate status Successful Completion.
3. Verify that the completion queue entries for all commands from the NVM Read Command Group sent by NVMe Host 2 indicate status Successful Completion.
4. Verify that the completion queue entries for all commands from the NVM Write Command Group sent by NVMe Host 2 while not a registrant of the namespace indicate status Reservation Conflict.
5. Verify that the completion queue entries for all commands from the NVM Read Command Group sent by NVMe Host 2 while a registrant of the namespace indicate status Successful Completion.

Case 4: **Exclusive Access - Registrants Only or Exclusive Access - All Registrants (M, OF-FYI) Dual Port Devices Only**

An Exclusive Access - Registrants Only or an Exclusive Access - All Registrants reservation disallows commands from both the NVM Write Command Group and NVM Read Command Group by any non-registrant of the namespace. Registrants of the namespace are still allowed to issue commands from either command group.

The difference between the Exclusive Access - Registrants Only and Exclusive Access - All Registrants reservation types is that all registrants are also considered reservation holders with the Write Exclusive - All Registrants type.

This test requires there to be at least two controllers with a shared namespace in the NVM Subsystem.

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable. If the DUT is a single port device then this test is not applicable.
2. Configure NVMe Host 1 to issue a Set Features command with the Host Identifier feature to NVMe Controller 1 in order to set its Host Identifier for that controller.
3. Configure NVMe Host 2 to issue a Set Features command with the Host Identifier feature to NVMe Controller 2 with a different Host Identifier than NVMe Host 1.
4. Configure NVMe Host 1 to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to NVMe Controller 1 for the shared namespace in order to make the host a registrant of that namespace.
5. Configure NVMe Host 1 to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the reservation key currently associated with NVMe Host 1 in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to Exclusive Access - Registrants Only or Exclusive Access - All Registrants to NVMe Controller 1 in order for NVMe Host 1 to acquire a reservation of the specified type.
6. Configure each host to issue each of the commands in the NVM Read Command Group above to their respective controllers.

7. Configure each host to issue each of the commands in the NVM Write Command Group above to their respective controllers.
8. Configure NVMe Host 2 to issue a Reservation Register command with the Register Reservation Key action and supplying a reservation key in the NRKEY field to NVMe Controller 2 for the shared namespace in order to make the host a registrant of that namespace.
9. Configure NVMe Host 2 to issue each of the commands in the NVM Read Command Group above to NVMe Controller 2.
10. Configure NVMe Host 2 to issue each of the commands in the NVM Write Command Group above to NVMe Controller 2.
11. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command.
2. Verify that the completion queue entries for all commands sent by NVMe Host 1 indicate status Successful Completion.
3. Verify that the completion queue entries for all commands sent by NVMe Host 2 while not a registrant of the namespace indicate status Reservation Conflict.
4. Verify that the completion queue entries for all commands sent by NVMe Host 2 while a registrant of the namespace indicate status Successful Completion.

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 2.67 – Reservation Notification Log Page (FYI, OF-FYI)

Purpose: To determine if an NVMe Controller properly uses the Reservation Notification Log Page.

References:

NVMe Specification 5.14.1.9.1, NVMe v1.3 ECN 001

Resource Requirements:

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

Last Modification: April 8, 2019

Discussion: The Reservation Notification log page reports one log page from a time ordered queue of reservation notification log pages, if available. A new Reservation Notification log page is created and added to the end of the queue of reservation notifications whenever an unmasked reservation notification occurs on any namespace that is attached to the controller. The Get Log Page command:

- returns a data buffer containing a log page corresponding to the oldest log page in the reservation notification queue (i.e., the log page containing the lowest Log Page Count field; accounting for wrapping); and
- removes that Reservation Notification log page from the queue.

If there are no available Reservation Notification log page entries when a Get Log command is issued, then an empty log page (i.e., all fields in the log page cleared to 0h) shall be returned.

Test Setup: See Appendix A.

Case 1: **Retrieve Log (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. Configure the Testing Station such that it is acting as 2 hosts accessing a shared namespace on the DUT.
3. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each shared namespace attached to the NVMe Controller:
        i. Configure at least 2 NVMe Hosts to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for a shared namespace in order to make the hosts registrants of that namespace.
        ii. Configure one NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to Write Exclusive or Exclusive Access in order for the host to acquire a reservation on the namespace with the reservation type.
        iii. Configure the same NVMe Host to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to the type of the reservation being released, and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field to the NVMe Controller in order to release the reservation held by the host.
        iv. Configure the same NVMe Host to issue a Reservation Report command to the NVMe Controller.
4. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations. The Reservation release command is expected to generate a Reservation Notification.

5. Repeat steps 1-3 a second time.
6. Perform a Get Log Page for the Reservation Notification Log Page.

Observable Results:

1. Verify that the Reservation Notification Log Page returned indicated the following:
    a. Log Page Count of 2
    b. Log Page Type of Reservation Released.
    c. The number of Available Log Pages was 1.
    d. Correct Namespace ID.

Case 2: **Empty Log (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. Configure the Testing Station such that it is acting as 2 hosts accessing a shared namespace on the DUT.
3. For each NVMe Controller in the NVM Subsystem:
    a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
    b. For each shared namespace attached to the NVMe Controller:
        i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
        ii. Configure the NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to the NVMe Controller in order for the host to acquire a reservation on the namespace with the reservation type.
        iii. Configure the NVMe Host to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to the type of the reservation being released, and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field to the NVMe Controller in order to release the reservation held by the host.
        iv. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
4. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations. The Reservation release command is expected to generate a Reservation Notification..
5. Perform a Get Log Page for the Reservation Notification Log Page.
6. Perform a second Get Log Page for the Reservation Notification Log Page.

Observable Results:

1. Verify that the second Reservation Notification Log Page was returned with all 0's, since the Log Page was empty.

Case 3: **Wrapped Log Count (FYI, OF-FYI)**

Test Procedure:

1. Check the ONCS field to determine of the controller supports reservations. If the controller does not support reservations then this test is not applicable.
2. Configure the Testing Station such that it is acting as 2 hosts accessing a shared namespace on the DUT.
3. For each NVMe Controller in the NVM Subsystem:

a. Configure the NVMe Host to issue a Set Features command with the Host Identifier feature to the NVMe Controller in order to set its Host Identifier for that controller.
b. For each shared namespace attached to the NVMe Controller:
    i. Configure the NVMe Host to issue a Reservation Register command with the Reservation Register Action (RREGA) field set to 000b (i.e. Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field to the NVMe Controller for the namespace in order to make the host a registrant of that namespace.
    ii. Configure the NVMe Host to issue a Reservation Acquire command, setting the Reservation Acquire Action (RACQA) field to 000b (Acquire), supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field, and setting the Reservation Type (RTYPE) field to a reservation type supported by the namespace to the NVMe Controller in order for the host to acquire a reservation on the namespace with the reservation type.
    iii. Configure the NVMe Host to issue a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e. Release), setting the Reservation Type (RTYPE) field to the type of the reservation being released, and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field to the NVMe Controller in order to release the reservation held by the host.
    iv. Configure the NVMe Host to issue a Reservation Report command to the NVMe Controller.
4. Perform a Reservation Release command with the RRELA field set to 001b (Clear) for all existing reservations. The Reservation release command is expected to generate a Reservation Notification.
5. Repeat steps 1-3 1000 times until the Reservation Notification Log Queue wraps.
6. Perform a Get Log Page for the Reservation Notification Log Page.

Observable Results:

1. If the DUT claims support for NVMe v1.4 or later, verify that if the Reservation Notification log 64 bit Log Page Count wrapped, it rolled over to a new value of 1. Otherwise verify that the Log Page Count incremented correctly.

Possible Problems: None known.

Group 6: 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 2.68 – Autonomous Power State Transitions Enabled (M)

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

References:

NVMe Specification 5.14.1.12, Fig 90, Fig 108, Fig 124

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 2.69 – Return from Non–Operational State (FYI)

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

References:

NVMe Specification 8.4.1

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)**

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)**

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 2.70 – Autonomous Power State Transition (M)

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

NVMe Specification 8.4.2, 5.14.1.12

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.

Test Procedure:

Case 1: Proper Structure
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.

Case 2: Controller and Power State Basis
The Autonomous Power State Transition Feature uses Command Dword 11 and specifies the attribute information in the data structure indicated in the Autonomous Power State Transition data structure consisting of 32 of the entries defined.

If a Get Features command is issued for this Feature, the attributes specified are returned in Dword 0 of the completion queue entry and the Autonomous Power State Transition data structure is returned in the data buffer for that command.

Test Procedure:

1. Configure the NVMe Host to issue a Set Features command for the Autonomous Power State Transition Feature with the Autonomous Power State Transition Enable (APSTE) bit cleared to '0'.
2. Configure the NVMe Host to issue a Get Features command for the Autonomous Power State Transition Feature to the NVMe Controller.
3. Configure the NVMe Host to issue a Set Features command for the Autonomous Power State Transition with the Autonomous Power State Transition Feature with APSTE bit set to '1'.
4. Configure the NVMe Host to issue a Get Features command for the Autonomous Power State Transition Feature to the NVMe Controller.
5. For each power state supported by the NVMe Controller:

a. Configure the NVMe Host to issue a Set Features command for the Autonomous Power State Transition Feature with APSTE bit set to '1' and, in the Autonomous Power State Transition data structure entry for that power state, specify a non-operational power state in the Idle Transition Power State (ITPS) field and a value of 100ms in the Idle Time Prior to Transition (ITPT) field.

b. Ensure that the NVMe Controller is idle long enough for the controller to autonomously transition to the power state and then configure the NVMe Host to issue a Get Features command for the Power Management Feature to the NVMe Controller to get the current power state of the NVMe Controller.

c. Configure the NVMe Host to issue a Set Features command for the Autonomous Power State Transition Feature with APSTE bit set to '1' and, in the Autonomous Power State Transition data structure entry for that power state, specify a non-operational power state in the ITPS field and a value of 0ms in the ITPT field.

d. After 10 seconds, configure the NVMe Host to issue a Get Features command for the Power Management Feature to the NVMe Controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. While Autonomous Power State Transition is disabled for a power state, verify that the controller does not autonomously transition from that power state.
3. While Autonomous Power State Transition is enabled for a power state, verify that the controller autonomously transitions to the configured power states after the configured idle time prior to transition period.

Case 3: Configurations

Test Procedure:

1. For each valid, non-operational power state, configure the NVMe Host to issue a Set Features command for the Autonomous Power State Transition Feature to the NVMe Controller specifying the non-operational power state in the Idle Transition Power State (ITPS) field for each entry in the Autonomous Power State Transition data structure and a value of 100ms in each Idle Time Prior to Transition (ITPT) field. Ensure that the Autonomous Power State Transition Enable (APSTE) field of the Set Features command is set to '1'.
2. Configure the NVMe Host to issue a Set Features command for the Autonomous Power State Transition Feature to the NVMe Controller specifying an operational power state to transition in the ITPS field.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.
2. Verify that the NVMe Controller successfully transitions to each non-operational power state.
3. Verify that the completion queue entry for the Set Features command with ITPS field set to an operational power state indicates status Invalid Field in Command.

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 2.71 – Power State Entrance Latency (M)

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

Reference:
NVMe Specification 8.4
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 2.72 – Power State Exit Latency (M)

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

Reference:
NVMe Specification 8.4

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 2.73 – Relative Read Throughput (M)

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

Reference:

NVMe Specification 8.4

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 2.74 – Relative Write Throughput (M)

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

Reference:

NVMe Specification 8.4

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 2.75 – Host Controlled Thermal Management (M)

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

Reference:

NVMe Specification 8.4

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 7: Namespace Management

Overview:

This section describes a method for performing conformance verification for NVMe products implementing NVMe Namespace Management features.

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 2.76 – Namespace Management Identify Command (M, OF-FYI)

Purpose: To determine if an NVMe Controller properly implements the features of the Identify command relating to namespace management.

References:

NVMe Specification 6.1.6, 8.12, NVMe v1.3 ECN 003

Resource Requirements:

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

Last Modification: May 19, 2020

Discussion: Additional CNS values were added to the Identify command relating to namespace management.

Test Setup: See Appendix A.

Case 1: **CNS 10h & 11h – Namespace Lists (M, OF-FYI)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 10h and CDW1.NSID 00h to the NVMe Controller Under Test in order to receive back a Namespace List containing all allocated namespaces.
3. For each namespace returned in the Namespace List from the previous step, configure the NVMe Host to issue an Identify command specifying CNS value 11h and setting CDW1.NSID to the namespace identifier of the namespace to the NVMe Controller Under Test in order to receive back an Identify Namespace data structure for the specified namespace.

Observable Results:

1. Verify that the requested data structures are posted to the memory buffer indicated in PRP Entry 1, PRP Entry 2, and Command Dword 10, and that a command completion queue entry is posted to the Admin Completion Queue for each command issued to the NVMe Controller Under Test.
2. Verify that all Identify Namespace data structures returned by the controller are not zero filled.
3. Verify that unused entries in the Namespace List are zero filled.

Case 2: **CNS 12h – Controller List – Controllers Attached to a Namespace (M, OF-FYI)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 02h and CDW1.NSID 00h to the NVMe Controller Under Test in order to receive back a Namespace List containing active namespaces.
3. For each namespace returned in the Namespace List from the previous step, configure the NVMe Host to issue an Identify command specifying CNS value 12h, CDW10.CNTID value 00h, and setting CDW1.NSID to the namespace identifier of the namespace to the NVMe Controller Under Test in order to receive back a Controller List containing the controller identifiers of all controllers attached to the namespace.

Observable Results:

1. Verify that the requested data structures are posted to the memory buffer indicated in PRP Entry 1, PRP Entry 2, and Command Dword 10, and that a command completion queue entry is posted to the Admin Completion Queue for each command issued to the NVMe Controller Under Test.

2. Verify that the controller identifier of the NVMe Controller Under Test is contained within each Controller List returned by the controller.
3. Verify that each Controller List contains valid values and that unused entries in the Controller List are zero filled.

Case 3: **CNS 13h – Controller List – All Controllers (M, OF-FYI)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 13h and CDW10.CNTID value 00h to the controller in order to receive back a Controller List containing the controller identifiers of all controllers in the NVM subsystem. 0h is a valid controller identifier.

Observable Results:

1. Verify that the requested data structure is posted to the memory buffer indicated in PRP Entry 1, PRP Entry 2, and Command Dword 10, and that a command completion queue entry is posted to the Admin Completion Queue.
2. Verify that the controller identifier of the NVMe Controller Under Test is contained within the Controller List returned by the controller.
3. Verify that the Controller List contains valid values and that unused entries in the Controller List are zero filled.

Case 4: **Common Namespace Data Structure (M, OF-FYI)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 00h and CDW1.NSID value FFFFFFFFh to the controller in order to receive back an Identify Namespace data structure that specifies capabilities that are common across namespaces.

Observable Results:

1. Verify that the requested data structure is posted to the memory buffer indicated in PRP Entry 1, PRP Entry 2, and Command Dword 10, and that a command completion queue entry is posted to the Admin Completion Queue.

Case 5: **NSID Uniqueness (M, OF-FYI)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 02h and CDW1.NSID value 00h to the controller in order to receive back a list of Namespace IDs. Repeat this setup until all NSIDs are retrieved. For some systems this may require more than 1 Identify Command with CNS=02h .
3. Check if all namespaces have unique NSIDs.
    a. If not all NSIDs are unique, check that the not-unique NSIDs are in fact shared namespaces by writing a pattern to the namespace through one controller, and then checking that that same pattern can be read through all other controller that are sharing that namespace.

Observable Results:

1. Verify that all namespaces have unique NSIDs.

Case 6: **Namespace management not supported (FYI, OF)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT supports Namespace Management then this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS value 00h and CDW1.NSID value FFFFFFFFh to the controller.

Observable Results:

1. Verify that the controller returned status “Invalid Namespace or Format” to the Identify command.

Possible Problems: None.

Test 2.77 – Namespace Management Command (M, OF-FYI)

Purpose: To determine if an NVMe Controller properly implements the Namespace Management command.

References:

NVMe Specification 6.1, 8.12, NVMe v1.3 ECN 002

Resource Requirements:

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

Last Modification: April 9, 2019

Discussion: The Namespace Management command is used for managing namespaces. It can be used for creating and deleting namespaces. The Select (SEL) field of Command Dword 10 is used to specify the type of operation for the Namespace Management command.

The Namespace Management command uses the PRP Entry 1, PRP Entry 2, and Dword 10 fields. All other command specific fields are reserved.

Test Setup: See Appendix A.

Case 1: **Namespace Creation – Exceed Number Supported (M, OF-FYI)**

The create operation creates a new namespace. The new namespace will not be attached to any controller. The Namespace Attachment command must be used to attach a new namespace to a controller if desired.

The data structure used for the created operation is defined in the NVMe Specification and has the same format as the Identify Namespace data structure. However, the host is only allowed to set the following fields in the data structure:

- Namespace Size (NSZE)
- Namespace Capacity (NCAP)
- Formatted LBA Size (FLBAS)
- End-to-end Data Protection Type Settings (DPS)
- Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC)

All other fields are reserved and shall be cleared to 0.

For the creation operation, the CDW1.NSID field is reserved and shall be cleared to 0. The controller selects the next available namespace identifier to use for the new namespace. The namespace identifier of the new namespace is returned in Dword 0 of the completion queue entry for the command.

If creation of a new namespace would cause the number of namespace to exceed the number of supported namespaces, then the controller shall return status Namespace Identifier Unavailable.

If the size of the new namespace exceeds the amount of available space on the device, then the controller shall return status Namespace Insufficient Capacity and the Command Specific Information field of the Error Information Log specifies the total amount of NVM capacity required to create the namespace in bytes.

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Check the FNA field Bit 0. If set to 1, ensure any namespaces created in the following steps are created with the same format as all other existing namespaces on the device, if any. If set to 0, the device supports format

on a namespace basis and it is not necessary to ensure that any namespaces created in the following steps are created with the same format as all other existing namespaces on the device..

3. Configure the NVMe Host to issue a Namespace Management command specifying Select field value 0h (Create) and valid values for the attached data structure to the NVMe Controller Under Test in order to create a new namespace.
4. Configure the NVMe Host to issue an Identify command specifying CNS field value 11h and CDW1.NSID value set to the namespace identifier of the newly created namespace to the NVMe Controller Under Test in order to receive back the Identify Namespace data structure for that namespace.
5. Configure the NVMe Host to issue Namespace Management commands specifying Select field value 0h (Create) and valid values for the attached data structure to the controller until the number of namespaces exceeds the number of namespaces supported by the NVM subsystem.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that a new inactive namespace is created and that the capabilities returned in the Identify Namespace data structure match the appropriate values set using the Namespace Management command.
3. Verify that the Namespace Identifier Unavailable status is returned when the number of namespaces exceeds the number of supported namespaces.

Case 2: **Namespace Deletion (M, OF-FYI)**

The delete operation deletes an existing namespace. As a side effect of the delete operation, the namespace is detached from any controller as it is no longer present in the system. Namespaces detached due to a delete operation will become an inactive namespace.

There is no data structure transferred for the delete operation.

The CDW1.NSID field specifies the namespace to delete.

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Configure the NVMe Host to issue a Namespace Management command specifying Select field value 1h (Delete) and CDW1.NSID field set to an active namespace in order to delete the specified namespace. If no Namespace exists, then the NVMe Host should first create a Namespace.
3. Configure the NVMe Host to issue an Identify command specifying CNS field value 02h and CDW1.NSID value 00h to the controller in order to receive back a Namespace List containing active namespace IDs attached to the controller.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the deleted namespace is not contained within the Namespace List.

Case 3: **Namespace Creation – Insufficient Capacity (M, OF-FYI)**

The create operation creates a new namespace. The new namespace will not be attached to any controller. The Namespace Attachment command must be used to attach a new namespace to a controller if desired.

The data structure used for the created operation is defined in the NVMe Specification and has the same format as the Identify Namespace data structure. However, the host is only allowed to set the following fields in the data structure:

- Namespace Size (NSZE)

- Namespace Capacity (NCAP)
- Formatted LBA Size (FLBAS)
- End-to-end Data Protection Type Settings (DPS)
- Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC)

All other fields are reserved and shall be cleared to 0.

For the creation operation, the CDW1.NSID field is reserved and shall be cleared to 0. The controller selects the next available namespace identifier to use for the new namespace. The namespace identifier of the new namespace is returned in Dword 0 of the completion queue entry for the command.

If creation of a new namespace would cause the number of namespace to exceed the number of supported namespaces, then the controller shall return status Namespace Identifier Unavailable.

If the size of the new namespace exceeds the amount of available space on the device, then the controller shall return status Namespace Insufficient Capacity and the Command Specific Information field of the Error Information Log specifies the total amount of NVM capacity required to create the namespace in bytes.

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Configure the NVMe Host to issue a Namespace Management command specifying Select field value 0h (Create) and valid values for the attached data structure to the NVMe Controller Under Test in order to create a new namespace.
3. Configure the NVMe Host to issue an Identify command specifying CNS field value 11h and CDW1.NSID value set to the namespace identifier of the newly created namespace to the NVMe Controller Under Test in order to receive back the Identify Namespace data structure for that namespace.
4. Configure the NVMe Host to issue Namespace Management commands specifying Select field value 0h (Create) and valid values for the attached data structure to the controller until the amount of available space on the device is exceeded.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that a new inactive namespace is created and that the capabilities returned in the Identify Namespace data structure match the appropriate values set using the Namespace Management command.
3. Verify that the Namespace Insufficient Capacity status is returned if the size of a new namespace would exceed the available space on the device.

Case 4: **Namespace Deletion when NSID=FFFFFFFFh(M, OF-FYI)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management, then this test is not applicable.
2. Configure the NVMe Host to issue an Identify command specifying CNS field value 02h and CDW1.NSID value 00h to the controller in order to receive back a Namespace List containing active namespace IDs attached to the controller. Verify that no valid Namespaces exist.
3. Perform a Namespace Management command with action of Delete, and NSID=FFFFFFFFh.

Observable Results:

1. Verify that after the completion of the Namespace Management command, the controller posts a completion queue entry to the appropriate Completion Queue indicating status of success for the command. DUT's claiming to support NVMe v1.3 or earlier may implement this functionality, but DUT's claiming to support NVMe v1.4 or higher must implement this functionality.

Possible Problems:

For Case 1, there may not be enough capacity remaining in the NVM subsystem or the number of namespace limit may already be reached which would prevent creation of a new namespace. In this case, the NVMe Host may delete an existing namespace to allow for creation of a new namespace.

For Case 2, there may be no active namespaces available for deletion. In this case, the NVMe Host may create and set up an active namespace in order to perform procedure steps which require such a namespace.

The NVMe Host should return the state of the device to its state from prior to running these test cases in order to prevent issues with other tests.

Test 2.78 – Namespace Attachment Command (M, OF-FYI)

Purpose: To determine if an NVMe Controller properly implements the Namespace Attachment command.

References:

NVMe Specification 6.1, 8.12

Resource Requirements:

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

Last Modification: March 2, 2016

Discussion: The Namespace Attachment command is used to attach and detach controllers from a namespace.

The Select field of command Dword 10 of the Namespace Attachment command is used to specify the operation of the command (attach or detach). The data structure used in the command is a 4096 byte Controller List which specifies the controllers that are to be attached or detached as part of the command. If the Controller List data structure transferred with the Namespace Attachment command is not properly formatted or is otherwise invalid then the command shall return status Controller List Invalid.

The Namespace Attachment command uses the Command Dword 10 field. All other command specific fields are reserved.

Test Setup: See Appendix A.

Case 1: **Namespace Attachment (M, OF-FYI)**

If a Namespace Attachment command is issued with the Controller Attach action selected and the namespace is already attached to a controller specified in the Controller List data structure transferred with the command, then the command shall return status Namespace Already Attached.

If a Namespace Attachment command is issued with the Controller Attach action selected and the namespace is private and already attached to another controller, then the namespace shall not be attached to any of the controllers specified in the Controller List data structure transferred with the command and the command shall return status Namespace Is Private.

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Configure the NVMe Host to issue a Namespace Attachment command, specifying Select field value 0h (Attach), CDW0.NSID field value of an inactive namespace, and the controller identifier of the NVMe Controller Under Test in the attached data structure, to the controller in order to attach the specified namespace to the controller.
3. Configure the NVMe Host to issue an Identify command specifying CNS field value 02h and CDW1.NSID value 00h to the NVMe Controller Under Test in order to receive back a Namespace List containing active namespace IDs attached to the controller.
4. Configure the NVMe Host to issue a Namespace Attachment command, specifying Select field value 0h (Attach), CDW0.NSID field value of the same namespace used in the previous step, and the controller identifier of the NVMe Controller in the attached data structure, to the controller.

Observable Results:

1. Verify that after the completion of the command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.

2. Verify that the namespace attached to the controller via the Namespace Attachment command is contained within the returned Namespace List to signify that the namespace was successfully attached.
3. Verify that the status of the second Namespace Attachment command is Namespace Already Attached.

Case 2: **Namespace Detachment (M,OF-FYI)**

When a namespace is detached from a controller it becomes an inactive namespace on that controller. Previously submitted but uncompleted or subsequently submitted commands to the affected namespace are handled by the controller as if they were issued to an inactive namespace.

If a Namespace Attachment command is issued with the Controller Detach action selected and the namespace is not attached to a controller specified in the Controller List data structure transferred with the command, then the command shall return status Namespace Not Attached.

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Configure the NVMe Host to issue a Namespace Attachment command, specifying Select field value 1h (Detach), CDW0.NSID field value of an inactive namespace , and the controller identifier the NVMe Controller Under Test in the attached data structure, to the controller in order to attempt to detach the specified namespace from the controller.
3. Configure the NVMe Host to issue a Namespace Attachment command, specifying Select field value 1h (Detach), CDW0.NSID field value of a namespace attached to the controller, and the controller identifier of the NVMe Controller Under Test in the attached data structure, to the controller in order to detach the specified namespace from the controller.
4. Configure the NVMe Host to issue an Identify command specifying CNS field value 02h and CDW1.NSID value 00h to the NVMe Controller Under Test in order to receive back a Namespace List containing active namespace IDs attached to the controller.

Observable Results:

1. Verify that after the completion of each command, the controller posts a completion queue entry to the appropriate Completion Queue indicating the status for the command.
2. Verify that the status of the first Namespace Attachment command is Namespace Not Attached.
3. Verify that the namespace identifier of the namespace which was detached from the controller via the Namespace Attachment command is not contained within the returned Namespace List to signify that the namespace was successfully detached from the NVMe Controller Under Test.

Case 3: **Namespace Attachment, Incorrect Command Set (FYI,OF-FYI)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Check Bit 43 of the CAP.CSS field. If this bit is set to 0 this test is not applicable.
3. Configure the NVMe Host to issue a Namespace Attachment command, specifying Select field value 0h (Attach), CDW0.NSID field value of an active namespace, to a controller that does not support the I/O Command Set for the Namespace being attached. Ensure that the command set is enabled on the controller via the I/O Command Set Profile feature.

Observable Results:

1. Verify that that the Namespace Attach command completes with a status of I/O Command Set Not Supported.

Case 4: **Namespace Attachment, Command Set not Enabled (FYI,OF-FYI)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management then this test is not applicable.
2. Check Bit 43 of the CAP.CSS field. If this bit is set to 0 this test is not applicable.
3. Configure the NVMe Host to issue a Namespace Attachment command, specifying Select field value 0h (Attach), CDW0.NSID field value of an active namespace, to a controller that supports the I/O Command Set for the Namespace being attached, but the Command Set is not enabled via the I/O Command Set profile feature.

Observable Results:

1. Verify that that the Namespace Attach command completes with a status of I/O Command Set Not Enabled.

Case 5: **MAXDNA Namespace Attachment Limit Exceeded (FYI, OF-FYI)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management, then this test is not applicable.
2. Check the MAXDNA field. If MAXDNA is set to 0 this test is not applicable.
3. Configure the NVMe Host to issue a Namespace Attachment command, specifying Select field value 0h (Attach), CDW0.NSID field value of an active namespace, in order to attach the specified namespace to the Domain.
4. Repeat the previous step, attaching Namespaces until the MAXDNA limit is exceeded.

Observable Results:

1. Verify that the final Namespace Attach command completes with Status Namespace Attachment Limit Exceeded.

Case 6: **MAXCNA Namespace Attachment Limit Exceeded (FYI, OF-FYI)**

Test Procedure:

1. Check the OACS field to determine if the DUT supports Namespace Management. If the DUT does not support Namespace Management, then this test is not applicable.
2. Check the MAXCNA field. If MAXCNA is set to 0 this test is not applicable.
3. Configure the NVMe Host to issue a Namespace Attachment command, specifying Select field value 0h (Attach), CDW0.NSID field value of an active namespace, in order to attach the specified namespace to the controller.
4. Repeat the previous step, attaching Namespaces until the MAXCNA limit is exceeded.

Observable Results:

1. Verify that the final Namespace Attach command completes with Status Namespace Attachment Limit Exceeded.

Possible Problems:

There may not be an inactive namespace in the NVM subsystem. In this case, the NVMe Host may create an inactive namespace in order to perform procedure steps which require such a namespace.

Group 8: System Bus Registers

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 2.79 – PCI Express Capability Registers (M)

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

References:

NVMe Specification 2.5

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:

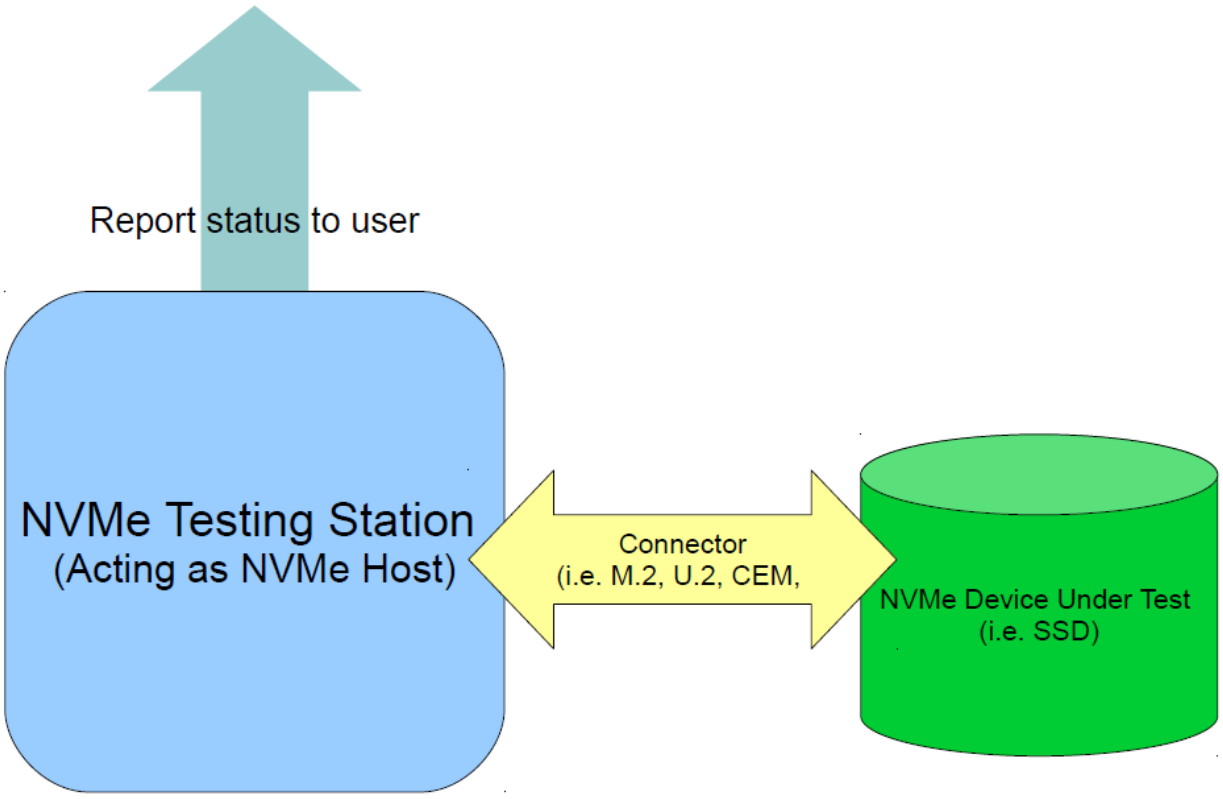
1. Verify that the PCIe Capabilities register (PXCAP) follows the format defined in section 2.5 of the NVMe Specification.
2. Verify that all reserved bits are set to 0.

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 v13.0 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 v13.0 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.