NVM Express™ Specifications: Mastering Today’s Architecture and Preparing for Tomorrow’s

J Metz, Cisco
Nick Adams, Intel
(with Special Guest, David Woolf!)

September 23-26, 2019
Santa Clara, CA
Agenda

▪ Current State of Standards
  ▪ To TP or Not TP (or, When is a TP not a Standard)?
▪ Why Refactor?
  ▪ Warts and All
  ▪ Incompatibilities
▪ Refactored NVMe™ Specification
▪ Compliance
What This Presentation Is/Is Not

- Is
  - Level-Setting
  - Communication of an architectural paradigm
  - Highly suggestive

- Is Not
  - Proscriptive
  - Exhaustive
Getting from Here to There
What's the process to get from “here” to “there”?

1. NVMe™ 1.2
2. NVMe 1.4
Well really...

NVMe™ 1.2

12 ECNs
14 Technical Proposals

NVMe 1.3

NVMe 1.4

6 ECNs
29 Technical Proposals
Well really…

**NVMe™ 1.2**
- 6 ECNs
- 29 Technical Proposals

**NVMe 1.3**
- This way: Easier Path
- That way? Dunno. Maybe bears. Wouldn’t risk it
- 1.3 ECNs + 1.4 ECNs
- 1.3 +1.4 Technical Proposals
- Mixed Test Cases
- + Refactoring

**NVMe 1.4**
- XX ECNs
- YY Technical Proposals
- ZZ Test Cases
- + Refactoring

**NVMe 2.0**
Use 1.4 as a Stepping Stone

More help from NVM Express going from 1.4 to 2.0
Preparing to Incorporate NVMe™ 1.4 Changes
Three Types of Changes Introduced

- New Features
- Feature Enhancements
- Required, Incompatible Changes
Where do I start?

- The NVM Express™ website, of course!
  - https://nvmexpress.org
  - Spec details at link: “Access Specification”
- Great resources
  - Current Spec
  - Current ECNs & TPs
  - Historical Specs
  - Detailed change documents

https://nvmexpress.org/changes-in-nvme-revision-1-4/
NVMe™ 1.4 Specification Required Changes*

- New NSID value usages
- New errors and reporting requirements
- Temperature threshold clarifications
- Controller Memory Buffer & Persistent Memory Region Enhancements
- New Sanitize requirements
- Reservation Notification Log usage
- Clarified LBA Range feature behavior
- Reservation Report command conflicts resolved
- New Abort command behavior

* Not to scale. These are categories of changes, not the full list of changes themselves
Example: Mandatory Change
Controller Memory Buffer (CMB)

- **Overview**
  - Controller Memory Buffer now requires Support (CMBS) and Enable (CRE) bit usage
  - Removed restrictions on the usages of the CMB – SQ, CQ & Data

- **Why the changes?**
  - Requires explicit configuration of the feature by the driver
  - Hardens the Controller Memory Buffer implementation
  - Relaxes the restrictions on host usage of the CMB

- **Impacts of inaction...**
  - Leaves the potential for DMA misrouting with CMB implementations

**References**
NVMe™ revision 1.4 section 3.1, 4.7, 4.8 & 7.3
Technical Proposal 4054
Example: Mandatory Change

**FFFFFFFF...udge This Noise**

- Overview – Namespace Identifiers
  - All usages of NSID value FFFFFFFF are now well-defined
  - Generally used to mean a broadcast action against all Namespaces

- What are the changes?
  - Clarifications in many sections: I/O Commands, Set/Get Features, Admin Commands, and Reservations
  - Explicitly defines when NSID of FFFFFFFF can be used and how to use it

- Why the changes?
  - The specification was quiet on a number of use cases
  - Need to provide consistency across Device and OS implementations
  - Improve the end-user experience and ease of NVMe device consumption

- Impacts of inaction
  - Inconsistent results when using devices from various hardware vendors
  - What happens when a Delete command is sent with NSID FFFFFFFFh? – More on that later…
New Feature Example: Persistent Event Log

- Persistent Event Log (optional)
  - Persistent capture of significant events for use by SW & system vendors that aren’t the device manufacturer
  - Defined Events:
    - Health Snapshot
    - Firmware Commits
    - Timestamp Changes
    - Power-on or Resets
    - Thermal Excursions
    - Vendor Specific
    - TCG-defined Events
    - Hardware Errors
    - Changed Namespace
    - Set Feature Events
    - Format NVM Start & Complete
    - Sanitize Start & Complete

- References
  - NVMe revision 1.4 section 5.14, 5.15, 5.21, 5.27 & 8.22
  - Technical Proposal 4007a, 4042a

Allows SSD customers to get consistent debug capabilities across vendors!

Allows SSD vendors an extensible framework for custom debug content!
Refactoring Philosophy
What Lessons Have We Learned?

- There are several places to go to look for information in the specifications
  - Maintaining consistency has been challenging
- PCIe is *not* the same as NVMe™, but there are times when it’s implied that they are
  - This is important because there is confusion generated in the marketplace, as a result
- Fabrics is arranged differently than the base spec
Architectural Trade-Offs

- What are the core elements of “what makes NVMe™?”
- What features will be adopted?
  - Optional features may or may not “take off”
  - What aspects are key and foundational?
  - Architectures should take these things into account
- Expect that there will be changes in command sets…
  - …Namespace types
  - …Transport methods
- Not about being proscriptive
  - It’s about recognizing what may change quickly, versus which become foundational
Refactoring NVMe™ Specification

Key Aspects Driving the Refactor

- Back to the core values… Fast, Simple, Scalable
- Foster areas of innovation while minimizing impact to broadly deployed solutions
- Creating an extensible spec infrastructure that will take the industry through the next phase of growth for NVMe!
Structuring for Extensibility

- Similar functions split between Base & Fabrics
- PCIe transport integrated into Base
- Command Sets not layered to enable extensibility
Structuring for Extensibility

- Similar functions split between Base & Fabrics
- PCIe transport integrated into Base
- Command Sets not layered to enable extensibility
Proposed Extensible Structure

NVMe™ Base Specification
- Current Base with removal of PCIe specifics
- Integrate NVMe-oF™ content
  - Identify, SGLs, Capsules, Discovery, ...

- Adds Fabrics concepts as core to NVMe
- Eliminates duplication in data structures
- Integration of NVMe and NVMe-oF base functions
- Separate command set specs
- Modular transport mapping layer, including PCIe
Compliance
Protecting Interoperability

- It’s not just about failing a compliance test. It’s about interoperability
  - For Phy signaling, users care about compliance for margin.
  - For protocol, users care about compliance as it affects interoperability.
  - Many developers are running protocol compliance checks nightly/weekly
- Let’s look at some protocol examples.
Compliance Test Cases

- Many tests take different paths depending upon which features are supported and which specification version is advertised.
- Host is going to pay attention to the version of the spec advertised and act differently.
Key Points on Compliance

▪ Refactoring (in and of itself) should not create more *tests*.
▪ Rather, refactoring means more test *documents*, as tests find new homes.
▪ Compliance to 1.4 spec will help enable a smooth migration to 2.0 compliance.
▪ Testing rubrics will become more involved as attention to interop and compliance becomes increasingly intertwined.
Spec Docs vs. Test Docs Today

- Today compliance program is focused on 3 specs: NVMe™ Base Spec, NVMe-MI™ Spec, NVMe-oF™ Spec. (Binding specs are in the queue).
  - Each has corresponding compliance test document
Spec Docs vs. Test Docs Tomorrow

- Refactoring can create new specs, which will require corresponding compliance test documents
- Existing tests may find new homes
  - E.g., Tests for “PCIe Binding” spec items currently reside in Base Spec Test Document, but will need to be migrated to a "PCIe Binding Spec Test Document"

NVMe Base Spec

NVMe-oF Spec

NVMe-MI Spec

NVMe/PCIe Transport Spec

Command Set Specs

NVMe/RDMA Transport Spec

NVMe/TCP Transport Spec
• New ECNs and TPs will create more tests, but refactoring should not.
• UNH-IOL is working on creating the correct test documents in a timely fashion
### Which Compliance Tests Apply to My Product?

**Today:**

<table>
<thead>
<tr>
<th>NVMe™/PCIe SSD</th>
<th>NVMe Base Spec Conformance Test Doc</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVMe/PCIe SSD with Management Interface (MI) support</td>
<td>NVMe Base Spec Conformance Test Doc</td>
</tr>
<tr>
<td>NVMe-oF™ AFA / JBOF etc…</td>
<td>NVMe Base Spec Conformance Test Doc</td>
</tr>
</tbody>
</table>

- **NVMe™/PCIe SSD**: 270 tests
- **NVMe/PCIe SSD with Management Interface (MI) support**: 323 tests
- **NVMe-oF™ AFA / JBOF etc…**: 217 tests
## Which Compliance Tests Apply to My Product?

**Tomorrow, in a Refactored World:**

<table>
<thead>
<tr>
<th>NVMe™/PCIe SSD</th>
<th>NVMe Base Spec Conformance Test Doc</th>
<th>NVMe/PCIe Binding Spec Conformance Test Doc</th>
<th>NVMe Block Command Set Spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>~270 tests</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NVMe/PCIe SSD with Management Interface (MI) support</th>
<th>NVMe Base Spec Conformance Test Doc</th>
<th>NVMe-MI Spec Conformance Test Doc</th>
<th>Command Set and Transport Specs</th>
</tr>
</thead>
<tbody>
<tr>
<td>~323 tests</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NVMe-oF™️ AFA / JBOF etc…</th>
<th>NVMe Base Spec Conformance Test Doc</th>
<th>NVMe/TCP Binding Spec Conformance Test Doc</th>
<th>NVMe Block Command Set Spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>~217 tests</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What could possibly go wrong?

- How does non-compliance, incorrect compliance, or lack of new features, affect correct operation and interoperability?
- (In other words, what can go wrong when things go wrong?) 🦖
Non-Compliance Ramification Example

- Get Feature Command for Namespace Specific Feature (i.e. Reservation Persistence) is sent with NSID=FFFFFFFFh
  - NVMe v1.4 Chapter 7.8
  - NVMe Base Spec Conformance Test 1.2 Case 7
- v1.3 behavior was that controller *may* accept. Error case undefined.
- v1.4 behavior is that products must return ‘Invalid Namespace or Format’.

<table>
<thead>
<tr>
<th>Host</th>
<th>NVMe v1.3</th>
<th>Controller</th>
<th>Host</th>
<th>NVMe v1.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Feature 83h NSID=FFFF FFFFh</td>
<td>NS1: Reservation Persistence = Persist through Power Loss?</td>
<td>Get Feature 83h NSID=02h</td>
<td>NS2: Reservation Persistence = No Persist through Power Loss</td>
<td></td>
</tr>
</tbody>
</table>
Non-Compliance Ramifications Example

- Namespace Management Command with Delete action and NSID=FFFFFFFFh
  - NVMe v1.4 Chapter 5.20
  - NVMe Base Spec Conformance Test 9.2 Case 4
- v1.3 behavior was that Delete action with NSID=FFFFFFFFh may delete all namespaces.
- v1.4 behavior was that Delete action with NSID=FFFFFFFFh deletes all namespaces.

NVMe v1.3

- Host
  - Identify CNS=02h
  - Active NSID List
  - NSID=0h

Controller
- NS List: NS1, NS2
- NS Mgmt
- Delete NSID=FFFFh

NVMe v1.4

- Host
  - Identify CNS=02h
  - Active NSID List
  - NSID=0h

Controller
- NS List: -
Compliance Summary

- Being aware of NVMe™ 2.0 architecture can help you to prepare today's NVMe 1.3 and NVMe 1.4 designs for migrating to v2.0
  - Rigorous compliance checking at NVMe 1.4 will smooth your transition to NVMe 2.0
  - Best way to prep for NVMe 2.0 compliance is to get NVMe 1.4 compliance right
Summary
At the end of the day...

- Changes to NVMe™ 1.4 specification are not just useful, but necessary
- It’s best not to wait to move to NVMe 1.4
  - More help from NVM Express™, Inc. for going from NVMe 1.4 to NVMe 2.0
- Changes in NVMe 2.0 specification will make it easier to find, develop and test
- Begin a NVMe 1.x -> NVMe 2.0 strategy plan ASAP
Backup
Breaking this down: Feature Enhancements

- **Enhanced Command Retry**
  - Defines enhancements to the command retry capability
    - Command Retry Delays: 3 different delay values or no delay
    - Error codes to indicate a command should be retried
    - Host discovery of support for the enhanced capabilities

- **Benefit**
  - Improved Host response to error conditions

- **References:**
  - NVMe™ revision 1.4 section 4.6, 5.15 & 5.21
  - Technical Proposal 4033
Benefits of Compliance for New Features

- **Enhanced Command Retry**
  - NVMe v1.4 Chapter 4.6, 5.15, 5.21,
  - NVMe Base Spec Conformance Test TBD
- v1.3 ‘Retry’ capability has one timer, and the controller can indicate if a command can or cannot be retried.
- v1.4 ‘Retry’ capability adds more timers, and the ability for controllers to indicate cannot, can, or should be retried.