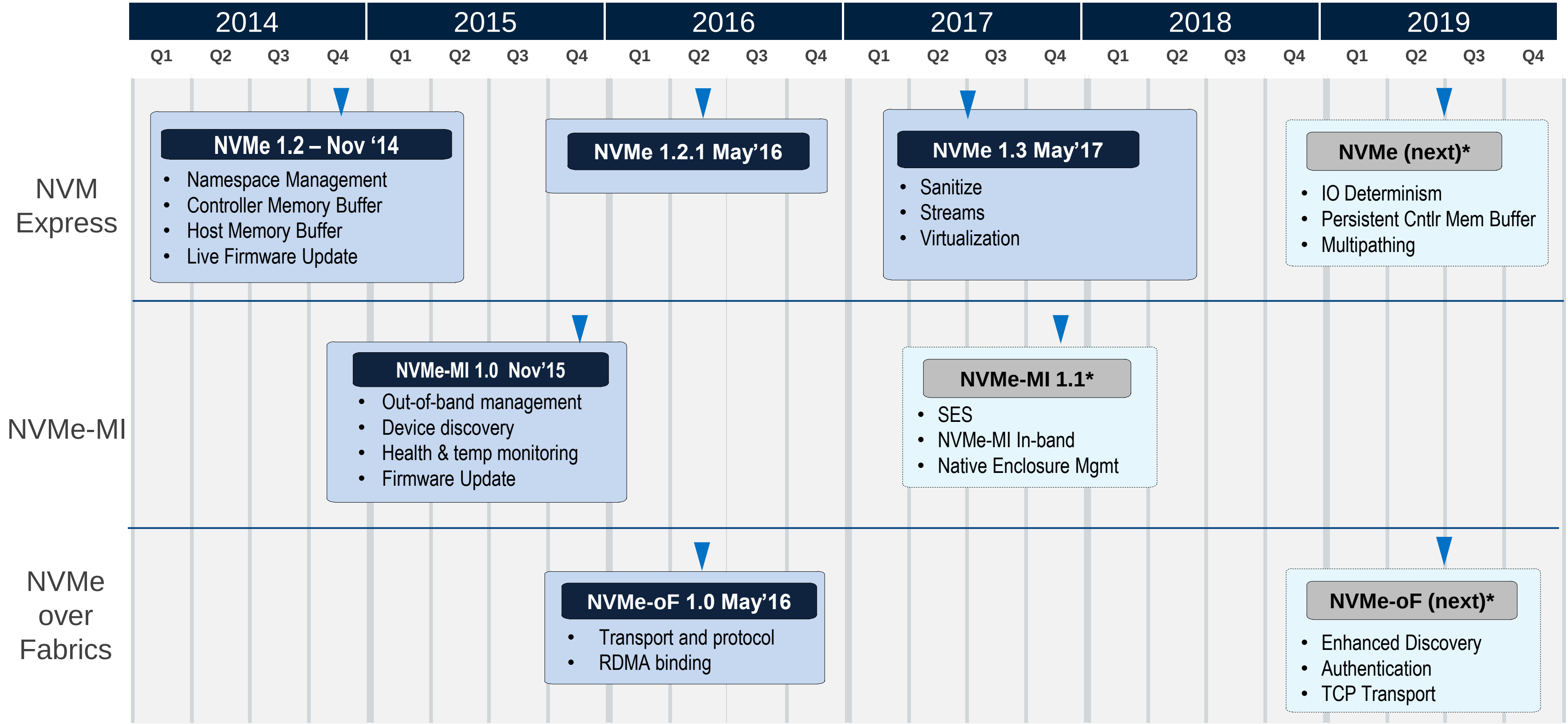
NVM Express 1.3 Delivering Continuous Innovation

June 2017

Jonmichael Hands, Product Marketing Manager Intel, NVM Express Marketing Co-Chair

View recorded webcast NVMe 1.3 - Learn What's New! at: https://www.brighttalk.com/webcast/12367/262451/nvme-1-3-learn-whats-new

NVM Express, Inc. Roadmap



2014 | 2015 | 2016 | 2017 | 2018 | 2019

Q1 Q2 Q3 Q4 (each year)

NVM Express

NVMe 1.2 – Nov ‘14
- Namespace Management
- Controller Memory Buffer
- Host Memory Buffer
- Live Firmware Update

NVMe 1.2.1 May’16

NVMe 1.3 May’17
- Sanitize
- Streams
- Virtualization

NVMe (next)*
- IO Determinism
- Persistent Cntlr Mem Buffer
- Multipathing

NVMe-MI

NVMe-MI 1.0 Nov’15
- Out-of-band management
- Device discovery
- Health & temp monitoring
- Firmware Update

NVMe-MI 1.1*
- SES
- NVMe-MI In-band
- Native Enclosure Mgmt

NVMe over Fabrics

NVMe-oF 1.0 May’16
- Transport and protocol
- RDMA binding

NVMe-oF (next)*
- Enhanced Discovery
- Authentication
- TCP Transport

🟦 Released NVMe ⬜ Planned NVMe Specification releases

* Subject to change

New Features / Technical Proposals in NVMe 1.3

	Type	Description	Benefit
	Client/Mobile	Boot Partitions	Enables bootstrapping of an SSD in a low resource environment
		Host Controlled Thermal Management	Host control to better regulate system thermals and device throttling
	Data Center/Enterprise	Directives	Enables exchange of meta data between device and host. First use is Streams to increase SSD endurance and performance
		Virtualization	Provides more flexibility with shared storage use cases and resource assignment, enabling developers to flexibly assign SSD resources to specific virtual machines
		Emulated Controller Optimization	Better performance for software defined NVMe controllers
	Debug	Timestamp	Start a timer and record time from host to controller via set and get features
		Error Log Updates	Error logging and debug, root cause problems faster
		Telemetry	Standard command to drop telemetry data, logs
	Management	Device Self-Test	Internal check of SSD health, ensure devices are operating as expected
		Sanitize	Simple, fast, native way to completely erase data in an SSD, allowing more options for secure SSD reuse or decommissioning
		Management Enhancements	Allows same management commands in or out-of-band
	Storage	SGL Dword Simplification	Simpler implementation

Device Self Test

Host system can request the storage device (SSD) do perform tests to ensure it is functioning properly

Short – less than 2 min

Long – will continue after reset (can send format or another DST to stop)

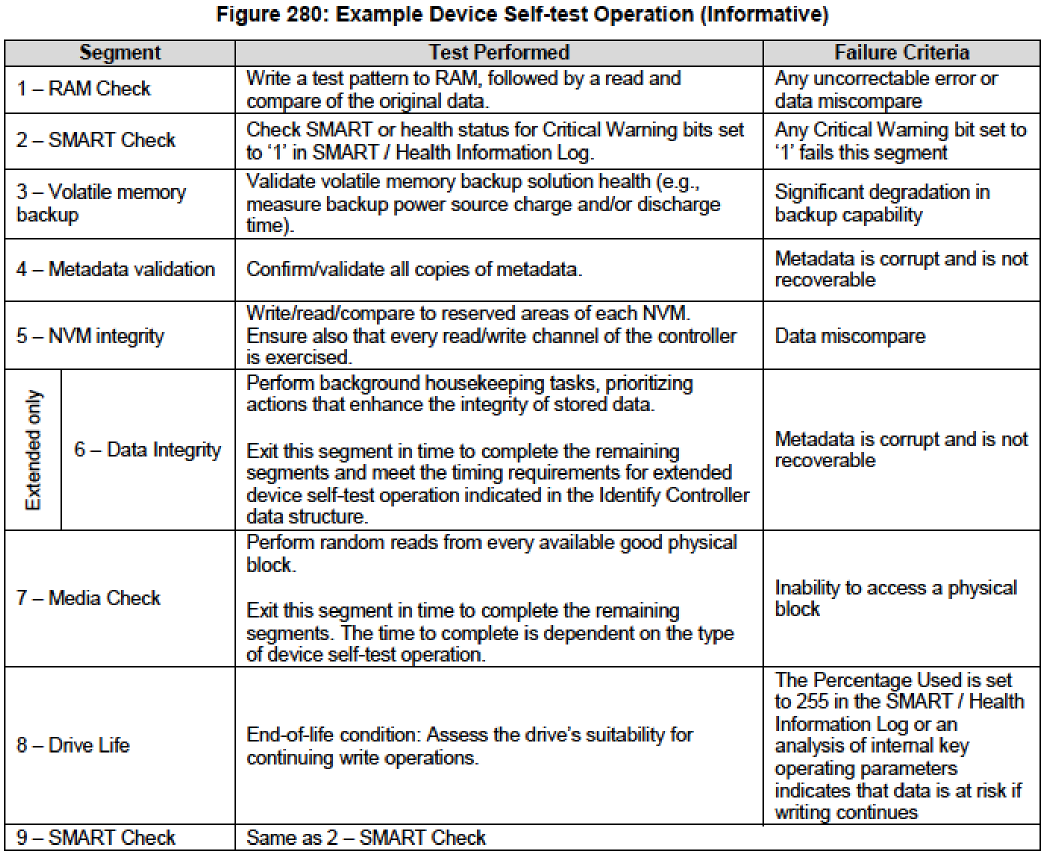


Figure 280: Example Device Self-test Operation (Informative)

Segment	Test Performed	Failure Criteria
1 – RAM Check	Write a test pattern to RAM, followed by a read and compare of the original data.	Any uncorrectable error or data miscompare
2 – SMART Check	Check SMART or health status for Critical Warning bits set to '1' in SMART / Health Information Log.	Any Critical Warning bit set to '1' fails this segment
3 – Volatile memory backup	Validate volatile memory backup solution health (e.g., measure backup power source charge and/or discharge time).	Significant degradation in backup capability
4 – Metadata validation	Confirm/validate all copies of metadata.	Metadata is corrupt and is not recoverable
5 – NVM integrity	Write/read/compare to reserved areas of each NVM. Ensure also that every read/write channel of the controller is exercised.	Data miscompare
Extended only 6 – Data Integrity	Perform background housekeeping tasks, prioritizing actions that enhance the integrity of stored data. Exit this segment in time to complete the remaining segments and meet the timing requirements for extended device self-test operation indicated in the Identify Controller data structure.	Metadata is corrupt and is not recoverable
7 – Media Check	Perform random reads from every available good physical block. Exit this segment in time to complete the remaining segments. The time to complete is dependent on the type of device self-test operation.	Inability to access a physical block
8 – Drive Life	End-of-life condition: Assess the drive's suitability for continuing write operations.	The Percentage Used is set to 255 in the SMART / Health Information Log or an analysis of internal key operating parameters indicates that data is at risk if writing continues
9 – SMART Check	Same as 2 – SMART Check	

Sanitize

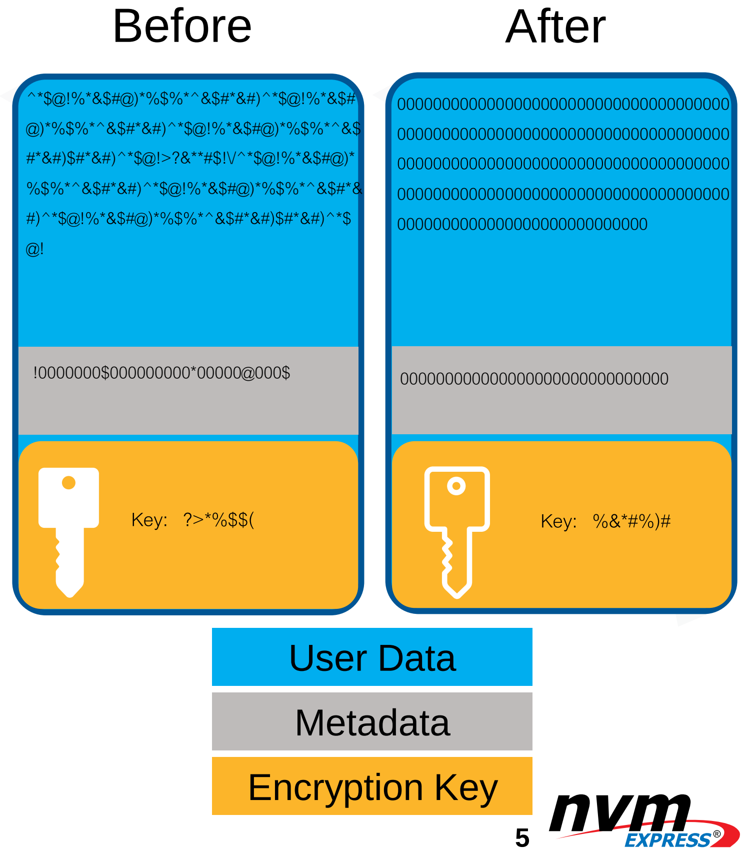
Alters user data so that is is unrecoverable by erasing media, metadata, and cache

Use when retiring SSD from use, reusing for new use case, or end of life

Modes in Sanitize

- Block Erase – low level block erase on media (physically erase NAND blocks)
- Crypto Erase - change media encryption key
- Overwrite – overwrite with data patterns (not good or recommended for NAND based SSDs due to endurance)

Sanitize vs Format Unit in NVMe – keeps going after reset, and erases all metadata, log pages and status during operation



Before

^*$@!%*&$#@)*%$%*^&$#*&#)^*$@!%*&$#
@)*%$%*^&$#*&#)^*$@!%*&$#@)*%$%*^&$
#*&#)$#*&#)^*$@!>?&**#$!\/^*$@!%*&$#@)*
%$%*^&$#*&#)^*$@!%*&$#@)*%$%*^&$#*&
#)^*$@!%*&$#@)*%$%*^&$#*&#)$#*&#)^*$
@!

!0000000$000000000*00000@000$

Key: ?>*%$$(

After

0000000000000000000000000000000000000
0000000000000000000000000000000000000
0000000000000000000000000000000000000
0000000000000000000000000000000000000
0000000000000000000000000000

000000000000000000000000000000

Key: %&*#%)#

User Data

Metadata

Encryption Key

New Debug Features

Timestamp

- Enables host to set a timestamp in controller via set features NVMe command, and read with get features

Error Log Updates

- Get Log NVMe command now returns more info on where the error occurred (queue, command, LBA, namespace, etc.) and error count

Telemetry

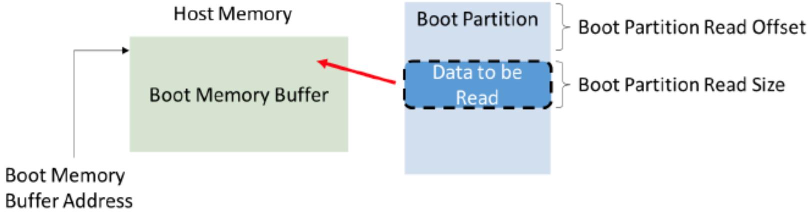
- vendor unique logs that can be dumped with industry standard commands and tools

Boot Partitions

- Optional storage area that can be read with “fast” initialization method (not standard NVMe queues). Example: UEFI bootloader
- Saves cost and space by removing the need for another storage medium (like SPI flash, EPROM)
- Write using standard NVMe Firmware Download and Firmware Commit
- Can be protected with **Replay Protected Memory Block**

Makes NVMe more accessible for mobile and client form factors

Figure 281: Boot Partition Overview



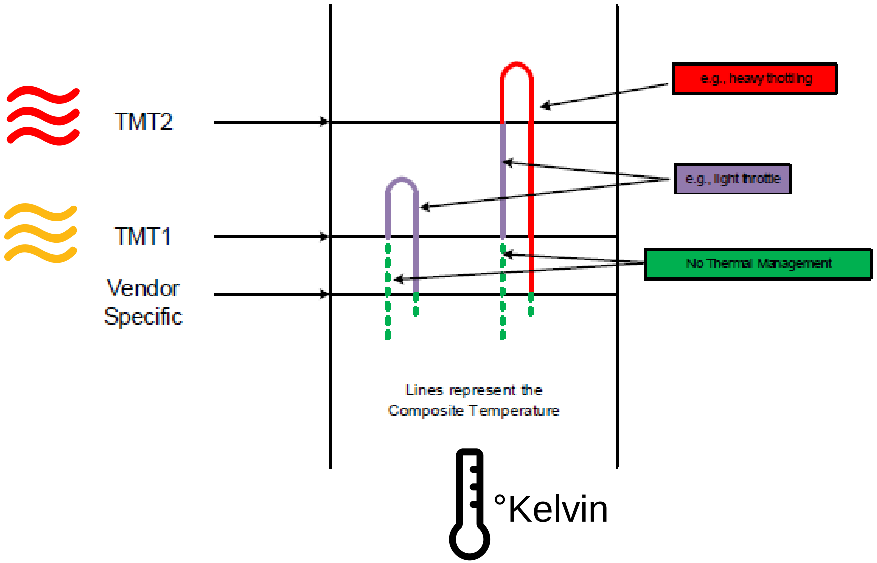
Host Controlled Thermal Management

Better thermal management in client systems like laptops and desktops.

Host can set **Thermal Management Temperature** at which a device should start going into a lower power state / throttling

- TMT1 – host tells SSD what temp in degrees K it should start throttling at
- TMT2 – threshold where the SSD should start heavy throttling regardless of impact to performance

Figure 264: HCTM Example



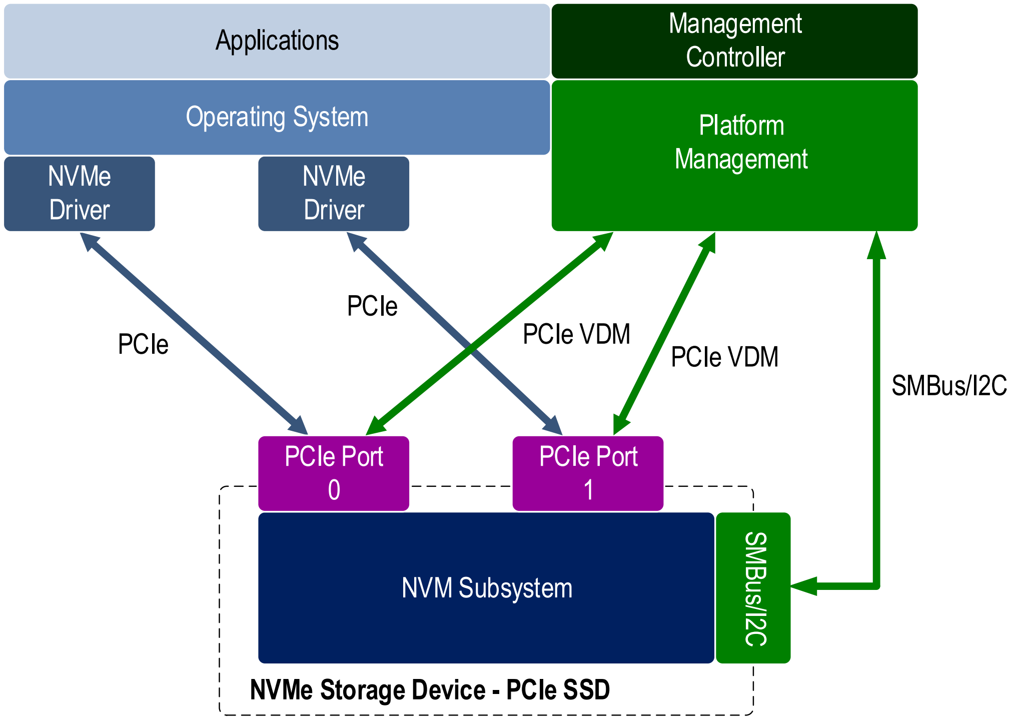
Management Enhancements

Management in-band: in operating system goes through NVMe admin queue

Examples: SMART, log pages, format unit

Management out-of-band: outside of host OS through SMBus/I2C or MCTP over PCIe

NVMe-MI in-band vs out-of-band



NVMe-MI Command Set Overview

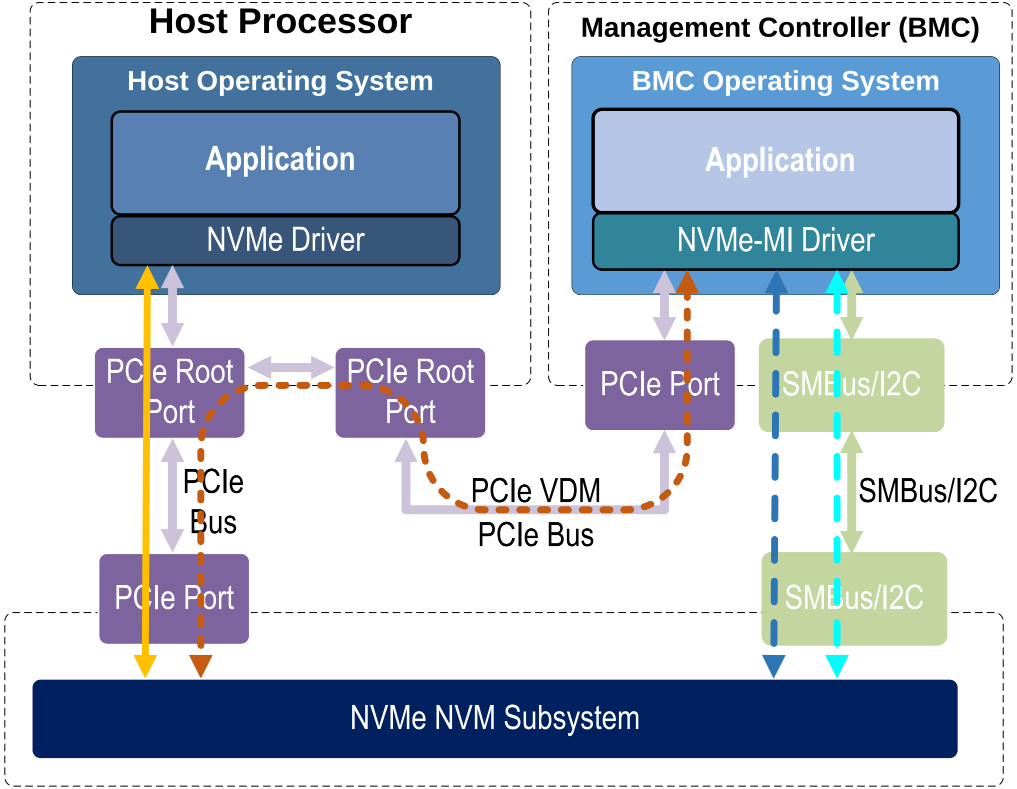
Command Type	Command
NVMe Management Interface Specific Commands	Read NVMe-MI Data Structure
	NVM Subsystem Health Status Poll
	Controller Health Status Poll
	Configuration Get
	Configuration Set
	VPD Read
	VPD Write
	Reset
	…
PCIe Command	PCIe Configuration Read
	PCIe Configuration write
	PCIe I/O Read
	PCIe I/O Write
	PCIe Memory Read
	PCIe Memory Write
	…

Command Type	Command
NVMe Commands	Firmware Activate/Commit
	Firmware Image Download
	Format NVM
	Get Features
	Get Log Page
	Identify
	Namespace Management
	Namespace Attachment
	Security Send
	Security Receive
	Set Features
	…

NVMe-MI Send / Receive Commands

Out-of-Band and In-band Data Flow

- Out-of-Band: NVMe-MI over MCTP over PCIe VDM
- Out-of-Band: NVMe-MI over MCTP over SMBus/I2C
- Out-of-Band: IPMI FRU Data Access (VPD) over SMBus/I2C
- In-Band: NVMe-MI Tunnel over NVMe

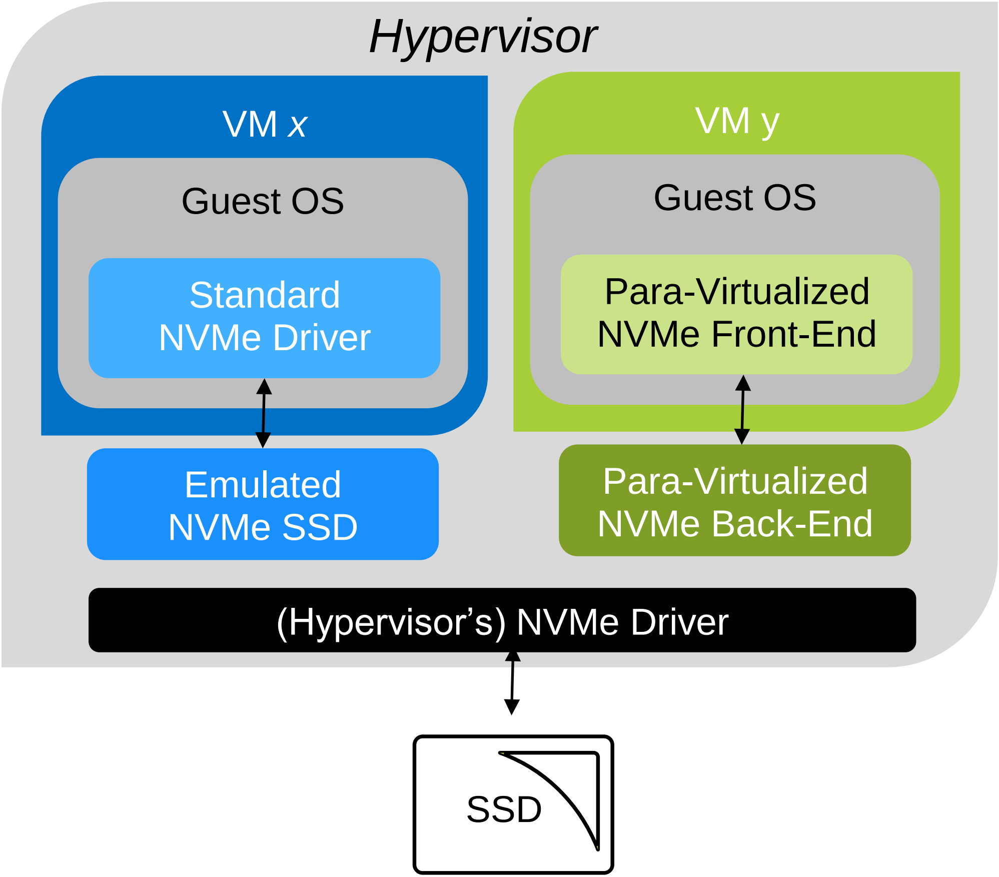


NVMe-MI 1.1 adds in-band NVMe-MI Tunnel

Storage Virtualization

Today’s virtualization model with NVMe uses software sharing

- Hypervisor Hardware Emulator is in the path of every IO
- Para-virtualized Drivers help reduce latency at the cost of using a non-standard NVMe driver

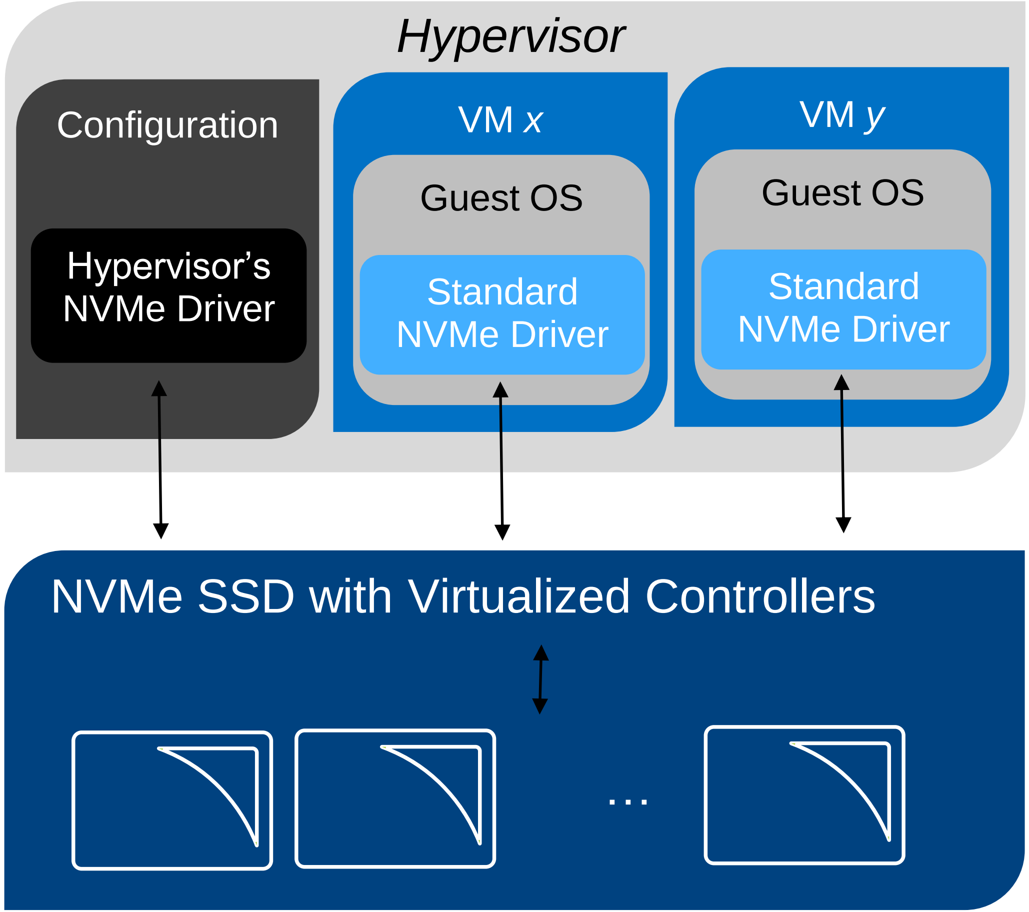


Hypervisor
VM *x*
Guest OS
Standard NVMe Driver
Emulated NVMe SSD
VM y
Guest OS
Para-Virtualized NVMe Front-End
Para-Virtualized NVMe Back-End
(Hypervisor’s) NVMe Driver
SSD

Virtualization Solution

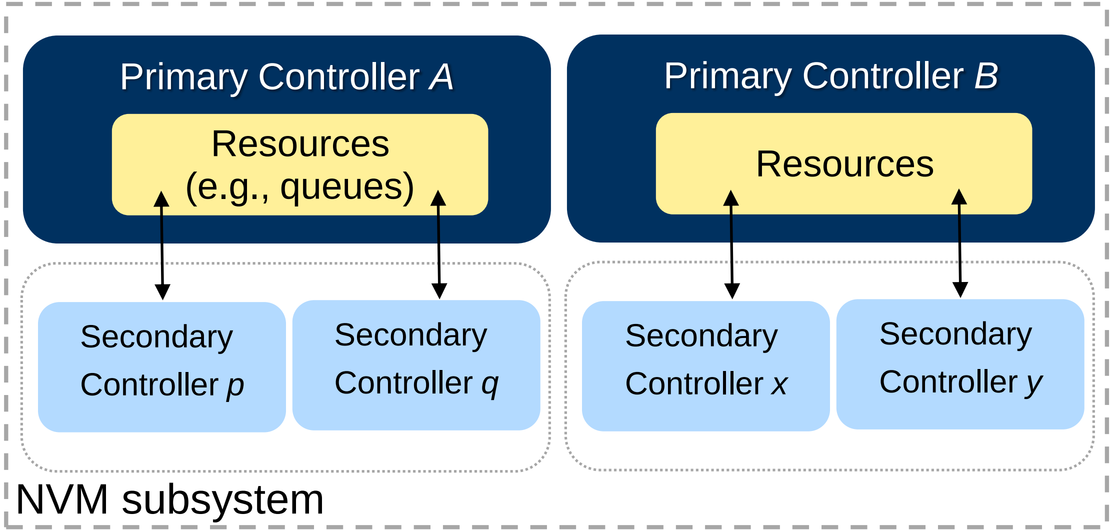
Direct Assignment

- Enable each tenant to "feel" like their portion of the SSD is a separate and distinct entity
- Hypervisor configures SSD – not involved in runtime access
- Guest OSes use today's standard NVMe drivers unmodified



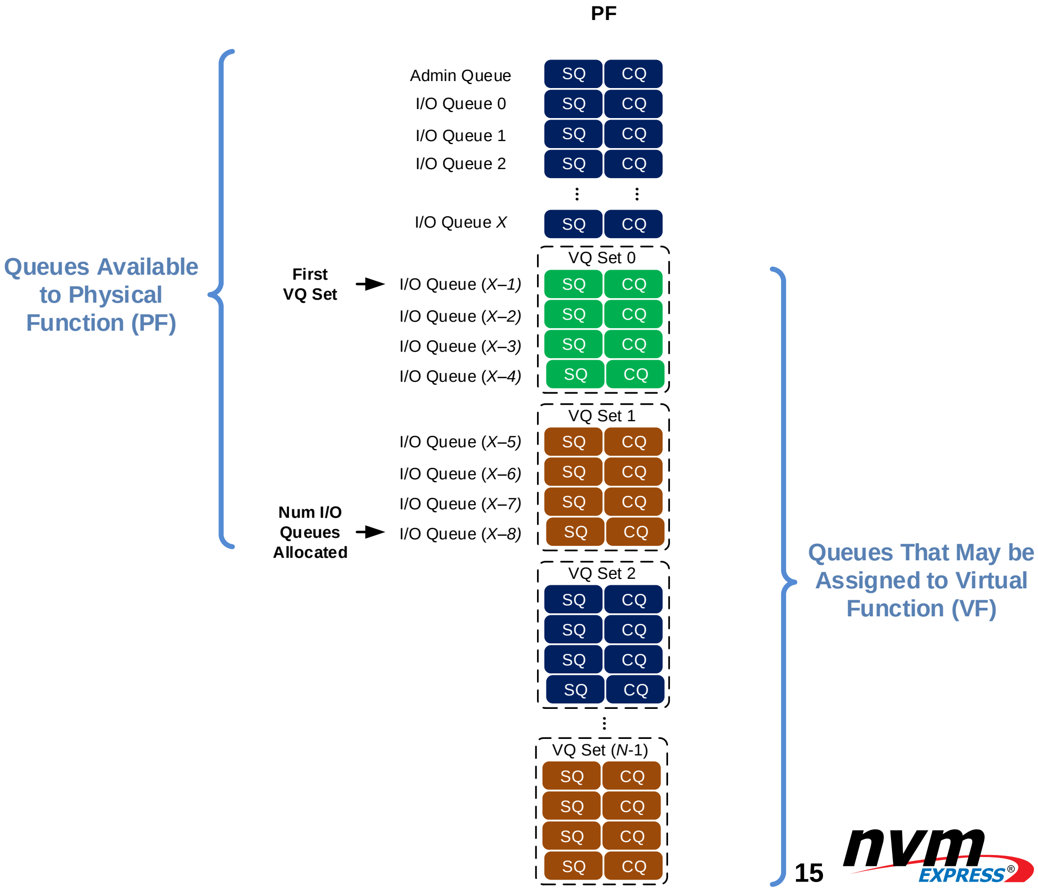
Direct Assignment in NVMe

- The near term approach maps onto PCIe SR-IOV
- There is a hierarchy of *primary* and *secondary* controllers
  - *primary* = physical function (PF)
  - *secondary* = virtual function (VF)
- Abstraction allows future mechanisms beyond SR-IOV



Allocating Resources

- Resources may be moved between the PF and VF(s)
- **VQ Set** – A set of (four) Submission Queue (SQ) and Completion Queue (CQ) pairs that may be assigned to a VF
- **VI Set** – A set of (four) MSI-X interrupt resources that may be assigned to a VF



Queues Available to Physical Function (PF)

Queues That May be Assigned to Virtual Function (VF)

Virtualization Enhancements

- Relies on Namespace Management
  - Namespaces divide the capacity of the drive
  - Namespaces allocated between Primary and Secondary Controllers
- Allocate Queue Resources between Primary and Secondary Controllers

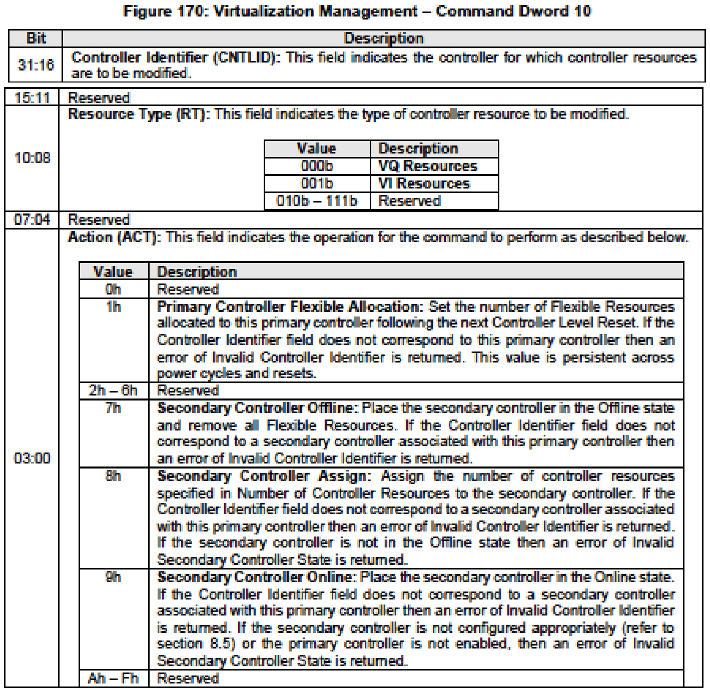


Figure 170: Virtualization Management – Command Dword 10

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31:16</td><td><b>Controller Identifier (CNTLID):</b> This field indicates the controller for which controller resources are to be modified.</td></tr>
<tr><td>15:11</td><td>Reserved</td></tr>
<tr><td>10:08</td><td><b>Resource Type (RT):</b> This field indicates the type of controller resource to be modified.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>000b</td><td><b>VQ Resources</b></td></tr>
<tr><td>001b</td><td><b>VI Resources</b></td></tr>
<tr><td>010b – 111b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>07:04</td><td>Reserved</td></tr>
<tr><td>03:00</td><td><b>Action (ACT):</b> This field indicates the operation for the command to perform as described below.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>Reserved</td></tr>
<tr><td>1h</td><td><b>Primary Controller Flexible Allocation:</b> Set the number of Flexible Resources allocated to this primary controller following the next Controller Level Reset. If the Controller Identifier field does not correspond to this primary controller then an error of Invalid Controller Identifier is returned. This value is persistent across power cycles and resets.</td></tr>
<tr><td>2h – 6h</td><td>Reserved</td></tr>
<tr><td>7h</td><td><b>Secondary Controller Offline:</b> Place the secondary controller in the Offline state and remove all Flexible Resources. If the Controller Identifier field does not correspond to a secondary controller associated with this primary controller then an error of Invalid Controller Identifier is returned.</td></tr>
<tr><td>8h</td><td><b>Secondary Controller Assign:</b> Assign the number of controller resources specified in Number of Controller Resources to the secondary controller. If the Controller Identifier field does not correspond to a secondary controller associated with this primary controller then an error of Invalid Controller Identifier is returned. If the secondary controller is not in the Offline state then an error of Invalid Secondary Controller State is returned.</td></tr>
<tr><td>9h</td><td><b>Secondary Controller Online:</b> Place the secondary controller in the Online state. If the Controller Identifier field does not correspond to a secondary controller associated with this primary controller then an error of Invalid Controller Identifier is returned. If the secondary controller is not configured appropriately (refer to section 8.5) or the primary controller is not enabled, then an error of Invalid Secondary Controller State is returned.</td></tr>
<tr><td>Ah – Fh</td><td>Reserved</td></tr>
</table></td></tr>
</table>

Directives

- A new framework in NVMe which enables per-IO command tagging and an admin capability to configure and report various settings and attributes
- Enables exchange of meta data between device and host

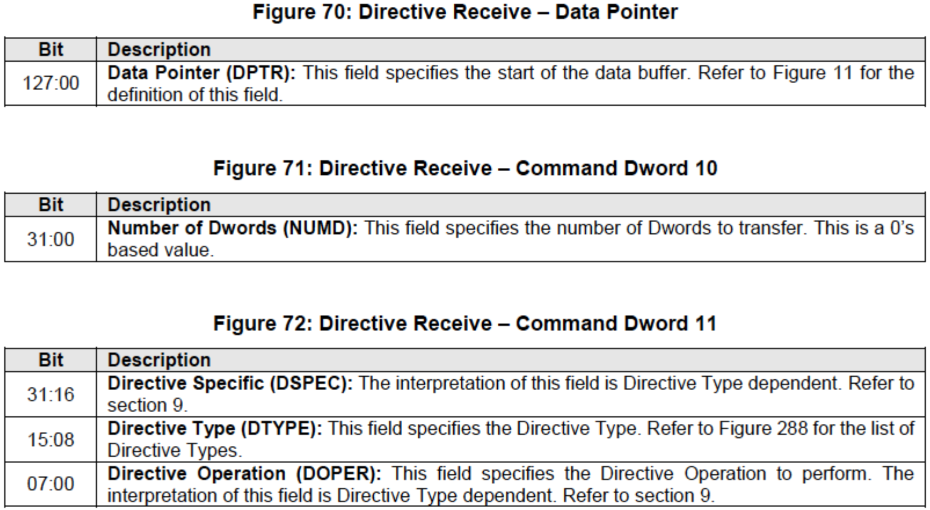


Figure 70: Directive Receive – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 for the definition of this field.

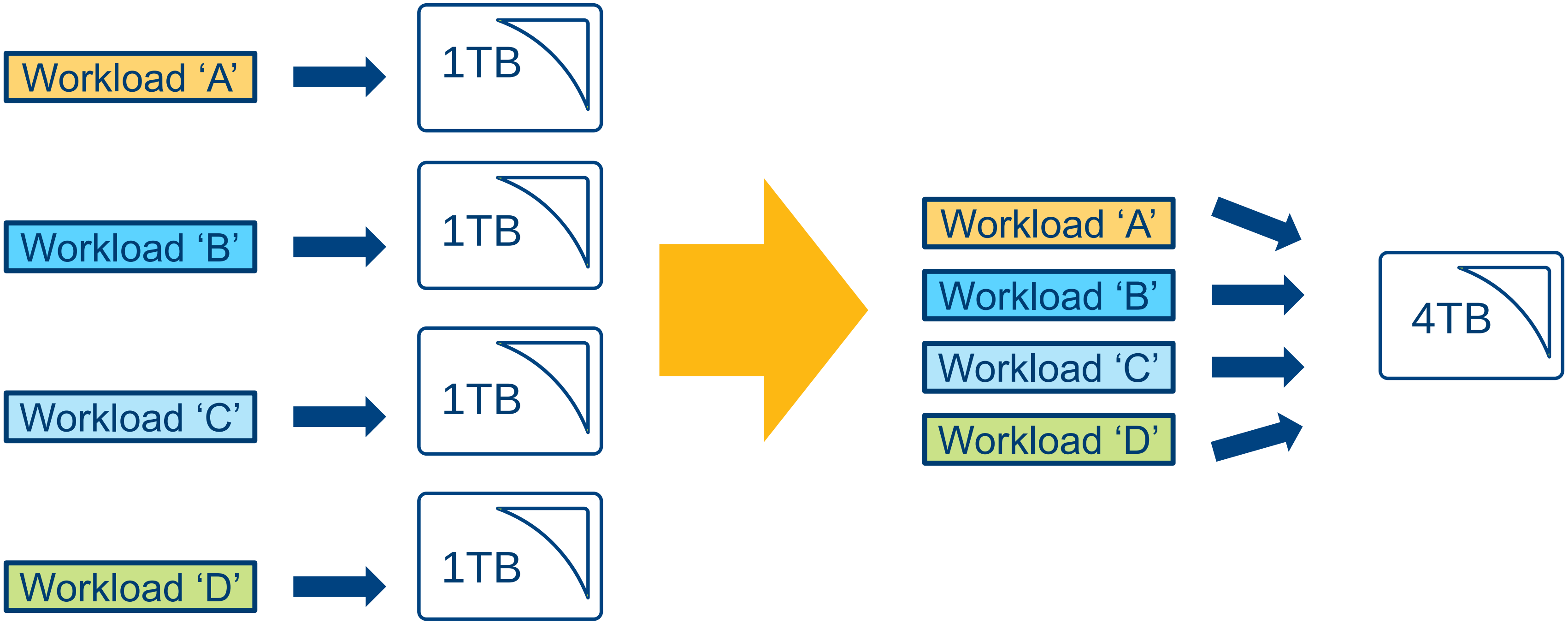
Figure 71: Directive Receive – Command Dword 10

Bit	Description
31:00	**Number of Dwords (NUMD):** This field specifies the number of Dwords to transfer. This is a 0's based value.

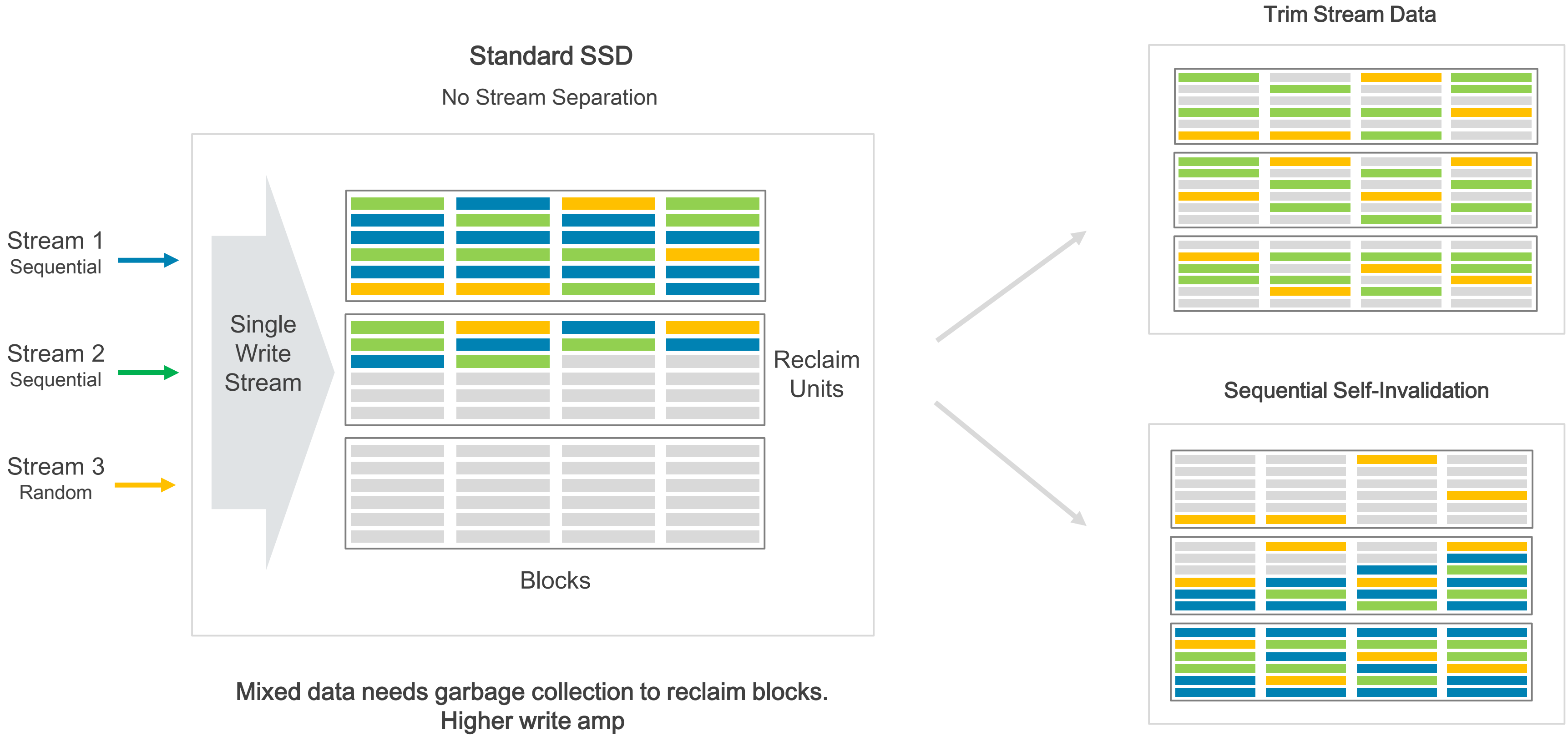
Figure 72: Directive Receive – Command Dword 11

Bit	Description
31:16	**Directive Specific (DSPEC):** The interpretation of this field is Directive Type dependent. Refer to section 9.
15:08	**Directive Type (DTYPE):** This field specifies the Directive Type. Refer to Figure 288 for the list of Directive Types.
07:00	**Directive Operation (DOPER):** This field specifies the Directive Operation to perform. The interpretation of this field is Directive Type dependent. Refer to section 9.

Streams: Problem



Streams: Problem



Trim Stream Data

Standard SSD

No Stream Separation

Stream 1
Sequential

Stream 2
Sequential

Stream 3
Random

Single Write Stream

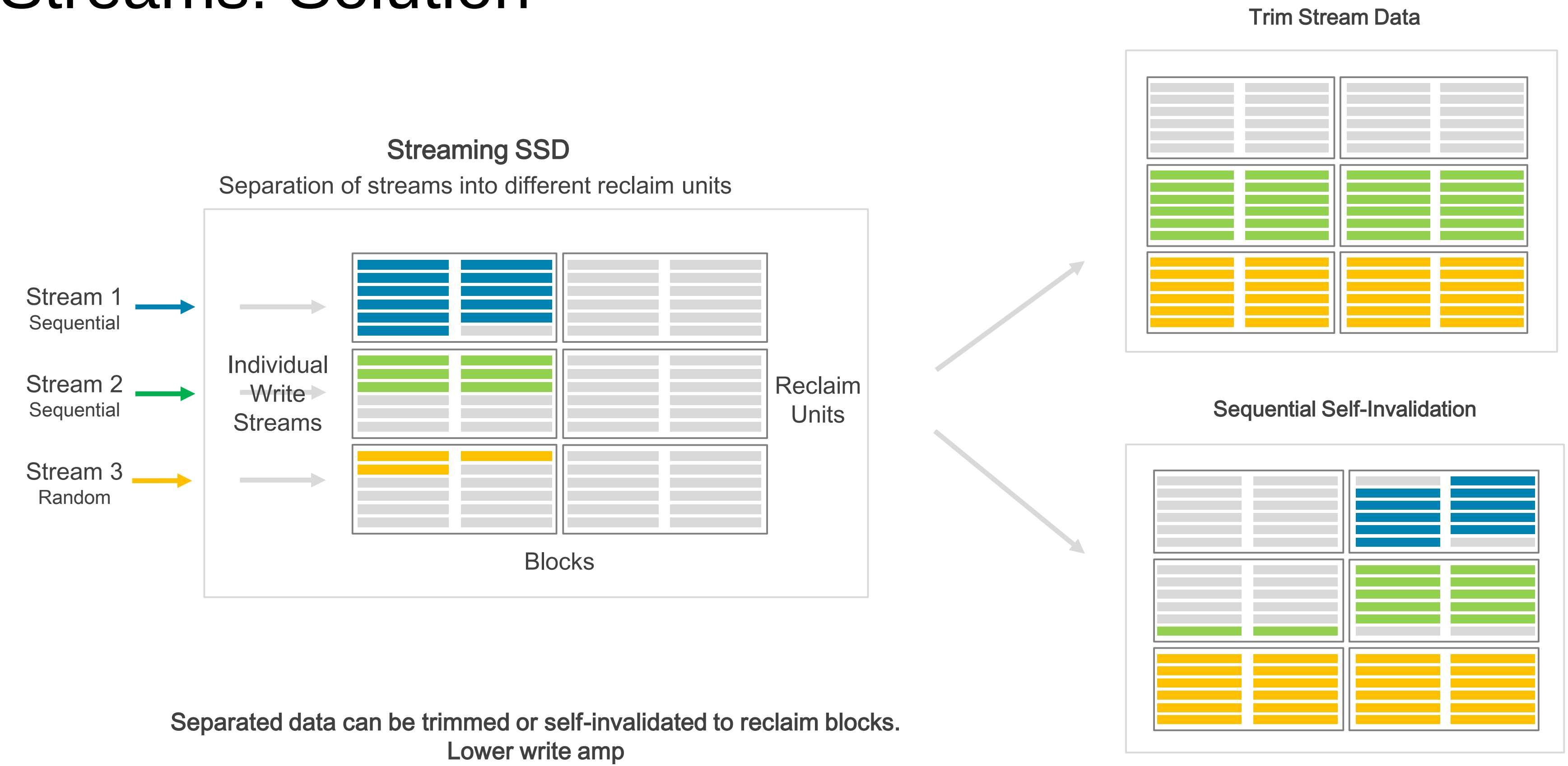
Reclaim Units

Blocks

Sequential Self-Invalidation

Mixed data needs garbage collection to reclaim blocks.
Higher write amp

Streams: Solution



Trim Stream Data

Streaming SSD

Separation of streams into different reclaim units

Stream 1
Sequential

Stream 2
Sequential

Stream 3
Random

Individual Write Streams

Reclaim Units

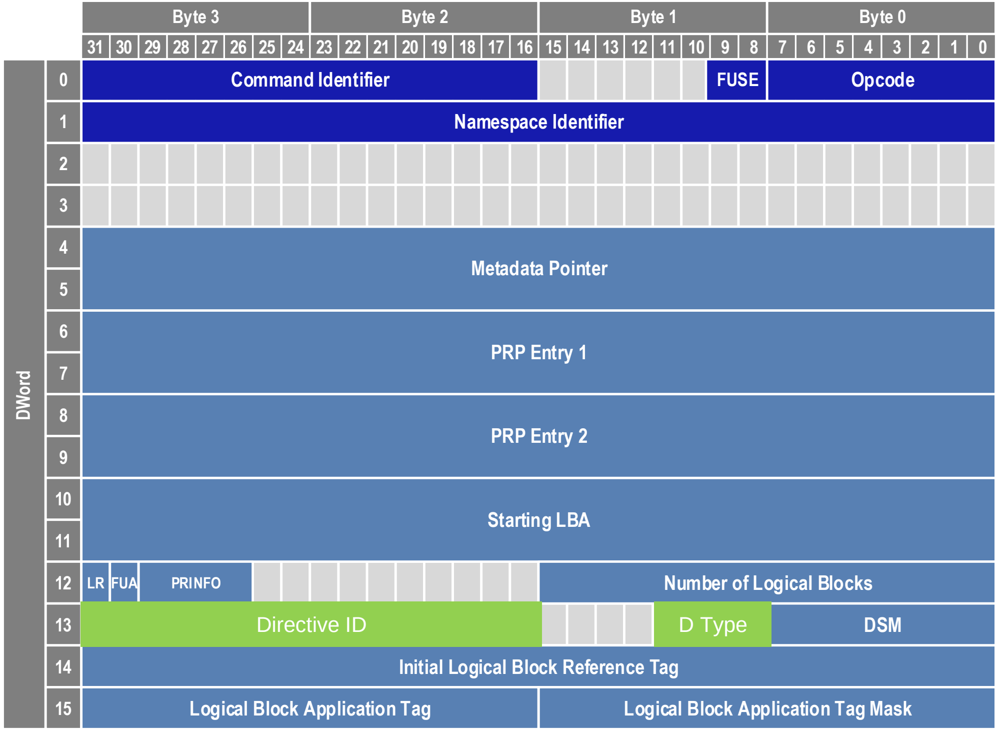
Blocks

Sequential Self-Invalidation

Separated data can be trimmed or self-invalidated to reclaim blocks.
Lower write amp

Enabling Future Enhancements

- Streams uses 16-bits in Write commands to identify stream
- NVMe commands have little available space …
- Make re-useable Directive ID / Directive Type field
- ID can be used for Streams today and future ideas tomorrow



DWord	Byte 3	Byte 2	Byte 1	Byte 0
	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0	Command Identifier		FUSE	Opcode
1	Namespace Identifier			
2				
3				
4	Metadata Pointer			
5				
6	PRP Entry 1			
7				
8	PRP Entry 2			
9				
10	Starting LBA			
11				
12	LR FUA PRINFO		Number of Logical Blocks	
13	Directive ID		D Type	DSM
14	Initial Logical Block Reference Tag			
15	Logical Block Application Tag		Logical Block Application Tag Mask	

nvm EXPRESS®

Architected for Performance