An NVM Express Tutorial

Kevin Marks

Dell, Inc.

What is NVM Express and Why

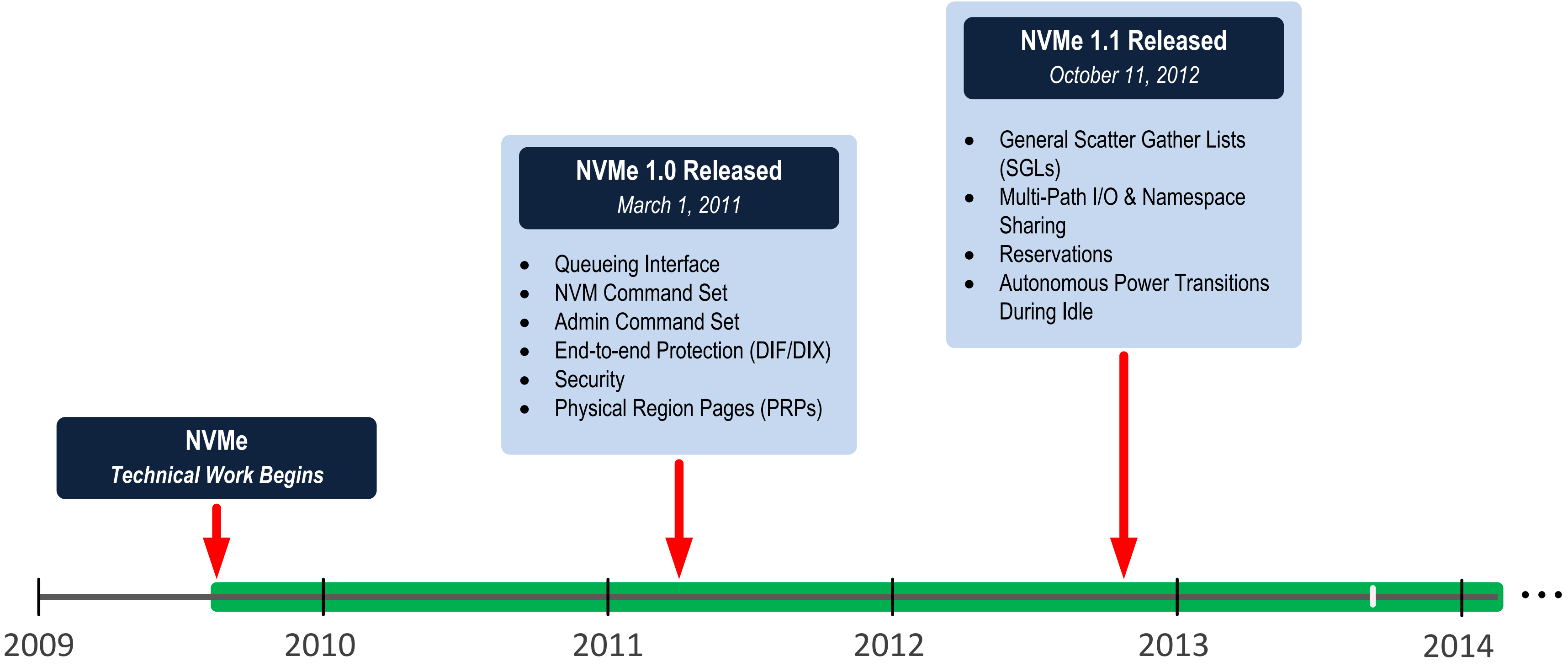
- NVM Express defines an optimized queuing interface, command set, and feature set for PCIe SSDs
  - Architected to scale from client to enterprise
- Standardization accelerates industry adoption
  - Standard drivers
  - Consistent feature set
  - Industry ecosystem
    - Development tools
    - Compliance and interoperability testing

Who created NVM Express (NVMe)

- NVM Express was developed by industry consortium of 90+ member companies and is directed by a 13-company Promoter Group

CISCO
DELL
EMC2
intel
NetApp
STEC
Micron
ORACLE
SanDisk
IDT
MARVELL
SAMSUNG
LSI

NVM Express Release Timeline



NVMe
Technical Work Begins

NVMe 1.0 Released
March 1, 2011

- Queueing Interface
- NVM Command Set
- Admin Command Set
- End-to-end Protection (DIF/DIX)
- Security
- Physical Region Pages (PRPs)

NVMe 1.1 Released
October 11, 2012

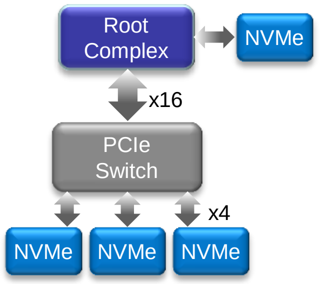
- General Scatter Gather Lists (SGLs)
- Multi-Path I/O & Namespace Sharing
- Reservations
- Autonomous Power Transitions During Idle

Goals of NVM Express relative to AHCI

- Remove uncacheable reads from command issue/completion
- Minimize MMIO writes in command issue/completion path
- Support for deep command queues and to simplify command decoding and processing
- Support MSI-X / flexible interrupt aggregation
- Support for many core systems
- Support Enterprise features
- Comprehensive statistics / Health status reporting / Robust error reporting & handling

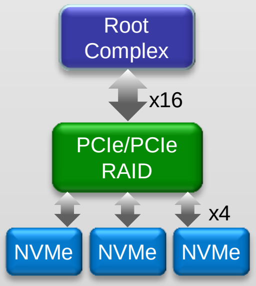
NVM Express Usage Models

Server Caching



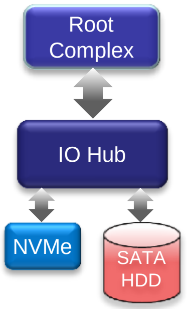
- Used for temporary data
- Non-redundant
- Used to reduce memory footprint

Server Storage



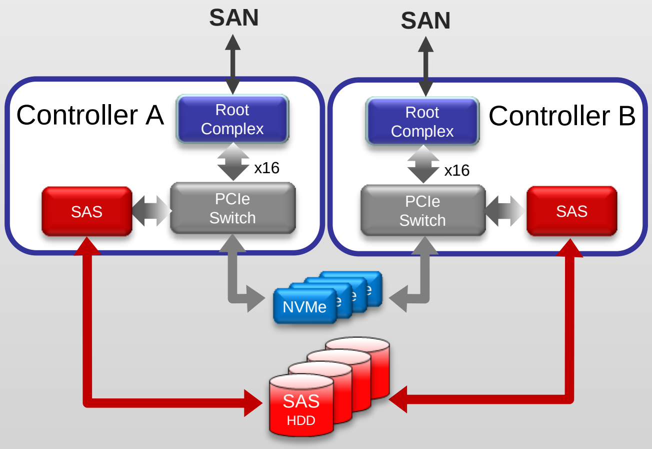
- Typically for persistent data
- Redundant (i.e., RAID'ed)
- Commonly used as Tier-0 storage

Client Storage



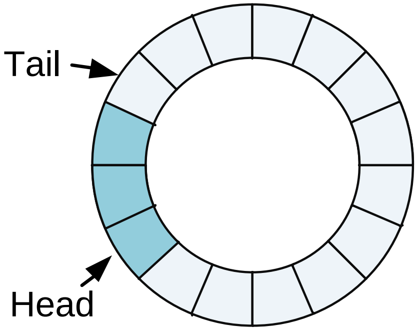
- Used for Boot/OS drive and/or HDD cache
- Non-redundant
- Power optimized

External Storage

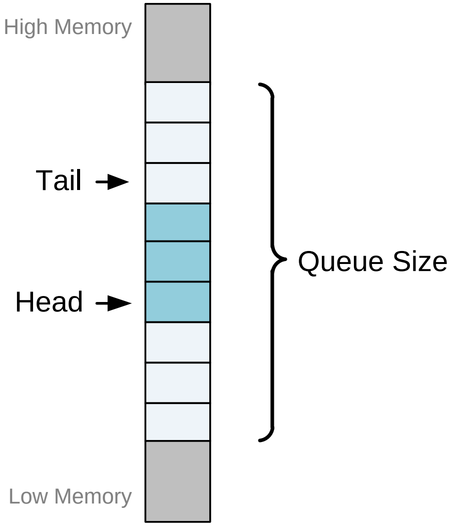


- Used for Metadata or data
- Multi-ported device
- Redundancy based on usage

NVMe Queues



Logical View



Physical View in Memory

- NVMe uses circular queues to pass messages (e.g., commands and command completion notifications.) The queues may be located anywhere in PCIe memory
  - Typically queues are located in host memory
  - Queues may consist of a contiguous block of physical memory or optionally a non-contiguous set of physical memory pages (defined by a PRP List)
- A Queue consists of set of fixed sized elements
- Tail
  - Points to next free element
  - If an element is added to the element pointed to by the tail, the tail is incremented to point to next free element taking wrapping into consideration
- Head
  - Points to next entry to be pulled off, if queue is not empty
  - If an element is removed from the element pointed to by the head, the head is incremented to point to the next element taking wrapping into consideration
- Queue Size (Usable)
  - Number of entries in the queue - 1
  - Minimum size is 2, Maximum is ~ 64K for I/O Queues and 4K for Admin Queue
- Queue Empty
  - Head == Tail
- Queue Full
  - Head == Tail + 1 mod # Of Queue Entries.

Types of Queues

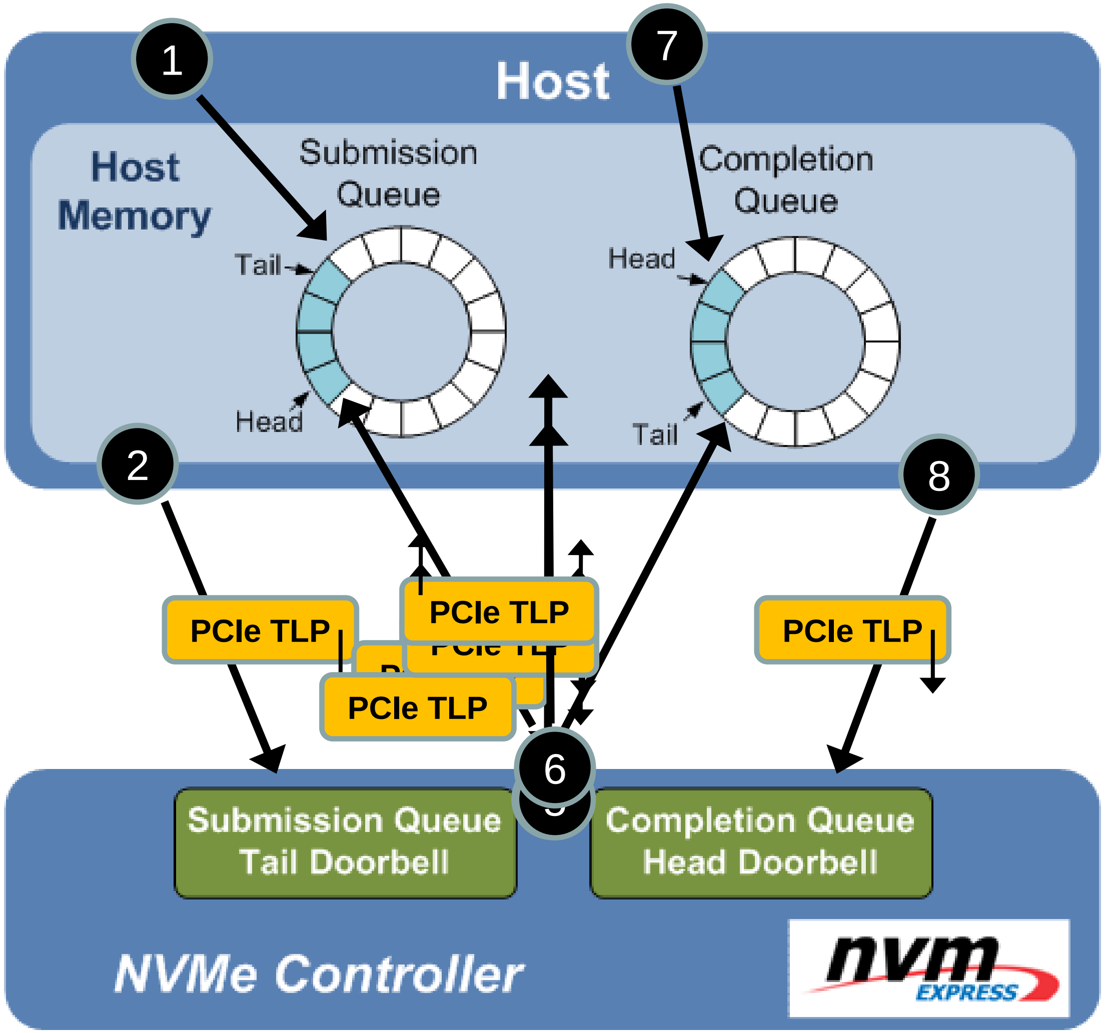
- Admin Queue for Admin Command Set
  - One per NVMe controller with up to 4K elements per queue
  - Used to configure IO Queues and controller/feature management
- I/O Queues for IO Command Sets (e.g., NVM command set)
  - Up to 64K queues per NVMe controller with up to 64K elements per queue
  - Used to submit/complete IO commands

Where each type has:

- Submission Queues (SQ)
  - Queues messages from host to controller
  - Used to submit commands
  - Identified by SQ ID
- Completion Queues (CQ)
  - Queues messages from controller to host
  - Used to post command completions
  - Identified by CQ ID
  - May have an independent MSI-X interrupt per completion queue

NVMe queues are messaging queues, not command queues

NVMe Command Execution

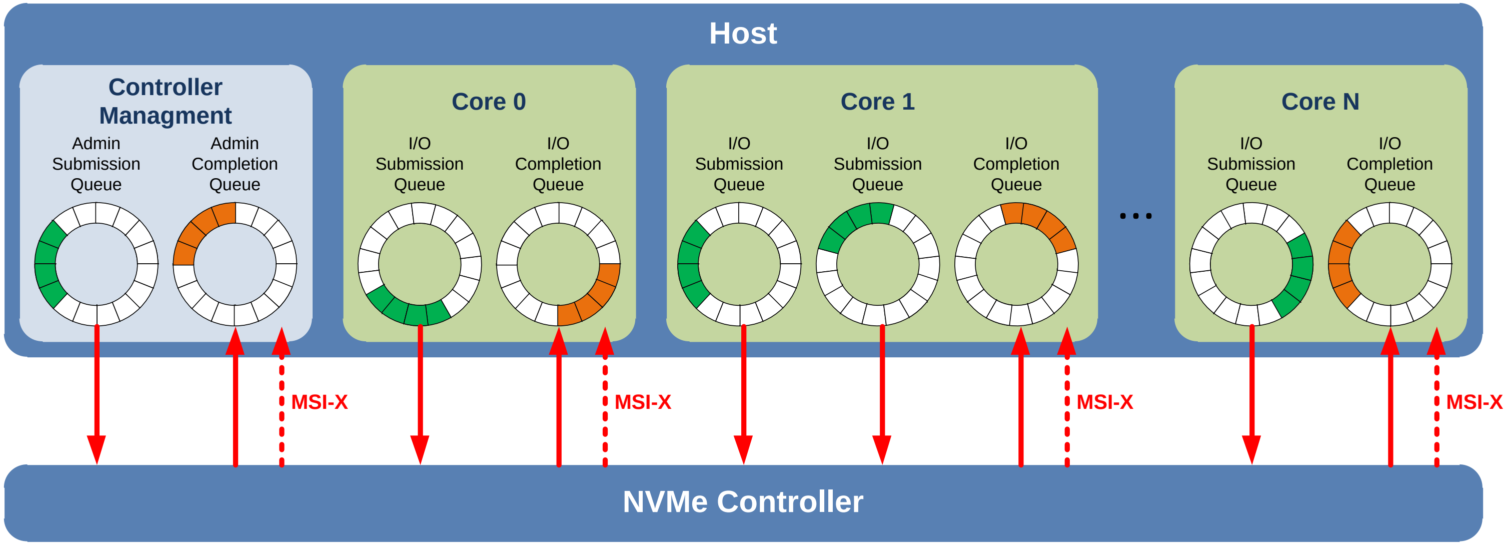


1) Queue Command(s)
2) Ring Doorbell *(New Tail)*
3) Fetch Command(s)
4) Process Command (s)
5) Queue Completion(s)
6) Generate Interrupt
7) Process Completion (s)
8) Ring Doorbell *(New Head)*

SQ and CQ relationships

- Each SQ is associated with only one CQ (i.e., commands submitted on a specific SQ complete on a specific CQ.
- The SQ to CQ relationship is defined at SQ creation time.
- It is permissible within the architecture to have multiple SQs mapped to a single CQ (n:1)

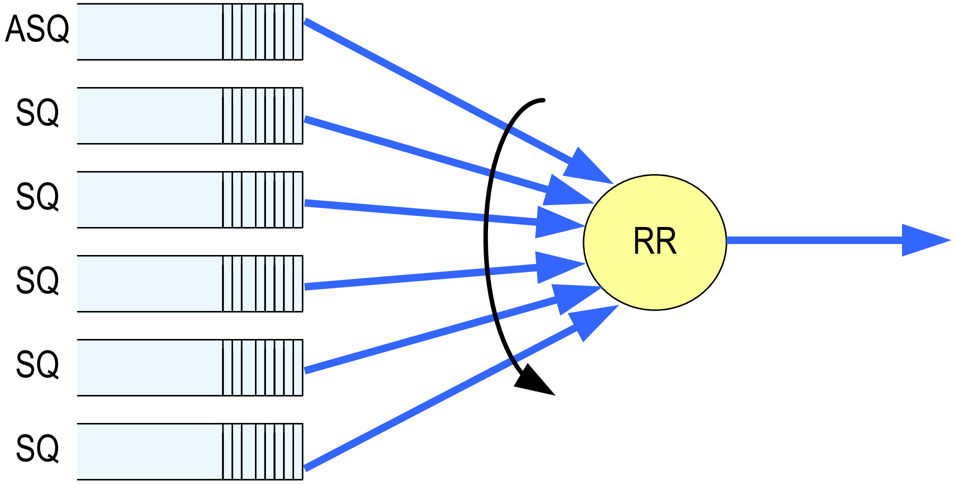
Scalable Queuing Interface



- Enables NUMA optimized drivers
  - Per core: One or more submission queues, one completion queue, and one MS-X interrupt
  - High performance and low latency command issue
  - No locking between cores
- Up to ~2^{32} outstanding commands
  - Support for up to ~ 64K I/O submission and completion queues
  - Each queue supports up to ~ 64K outstanding commands

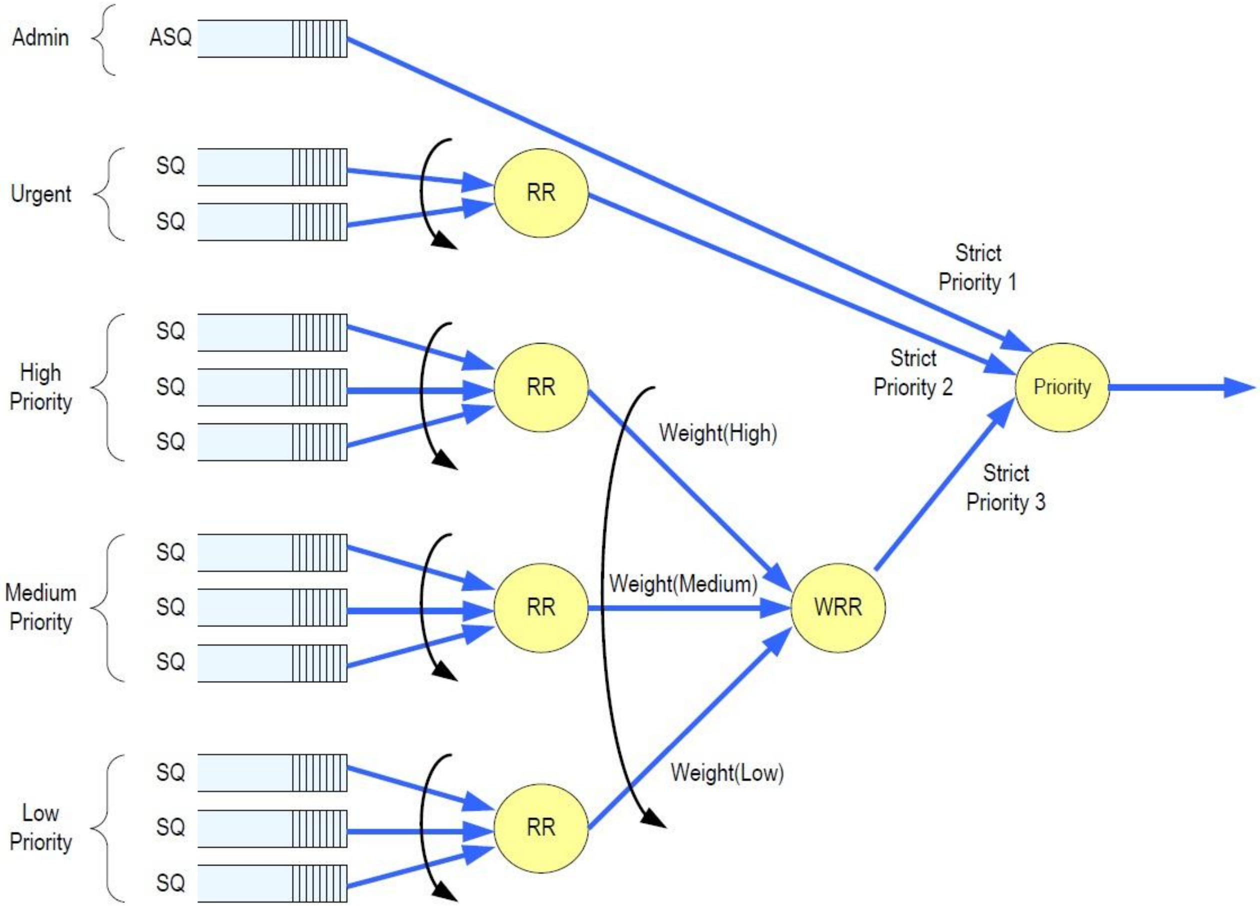
Command Arbitration

- All controllers support round robin arbitration

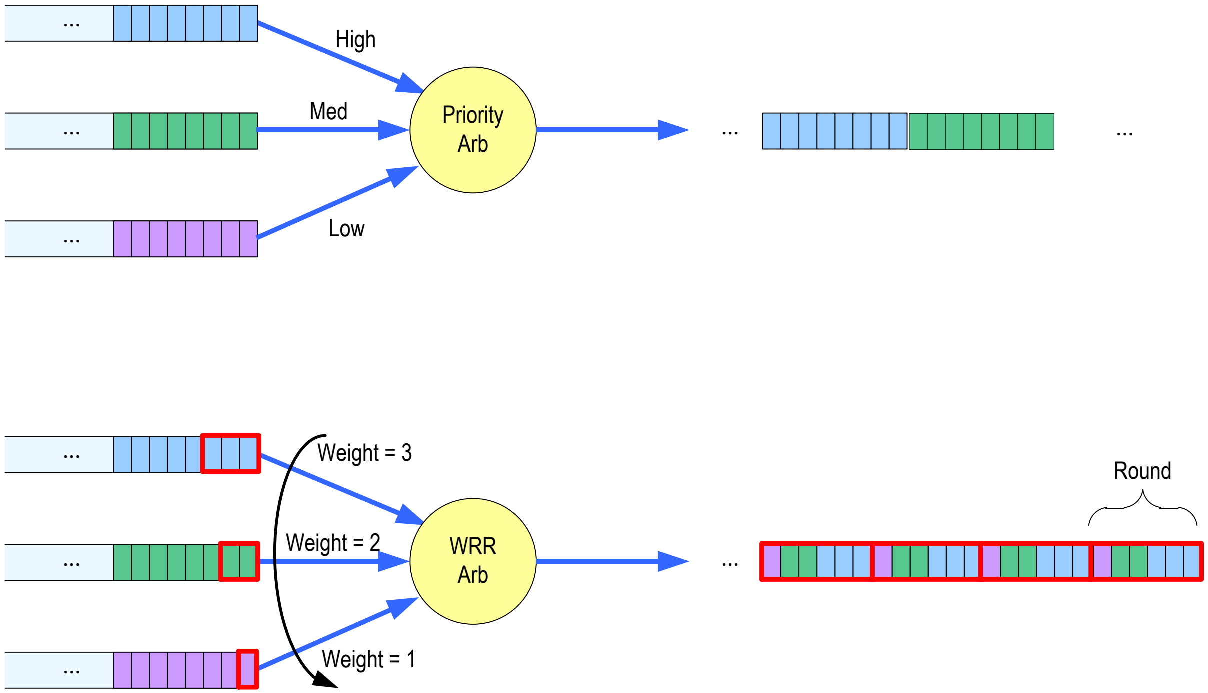


Command Arbitration

- An NVMe controller may support weighted round robin with urgent priority class arbitration



Arbitration Primitives



- Example above shown with an arbitration burst of no limit
- NVMe supports an arbitration burst of 1, 2, 4, 8, 16, 32, 64 and no limit
- NVMe supports 8-bit WRR weights

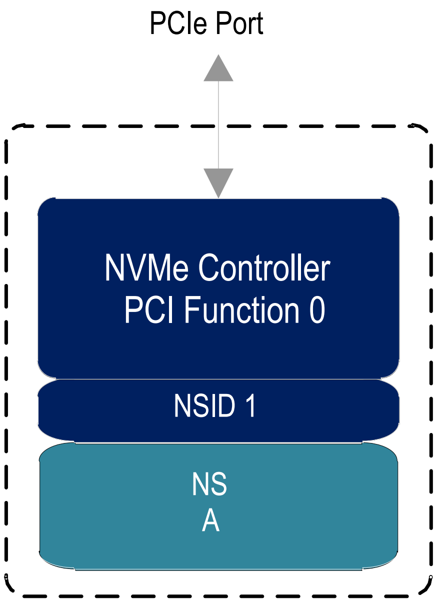
NVMe Subsystem Model

- **NVM Subsystem** - one or more controllers, one or more namespaces, one or more PCI Express ports, a non-volatile memory storage medium, and an interface between the controller(s) and non-volatile memory storage medium

- **Controller** – A PCI Express function that implements NVM Express

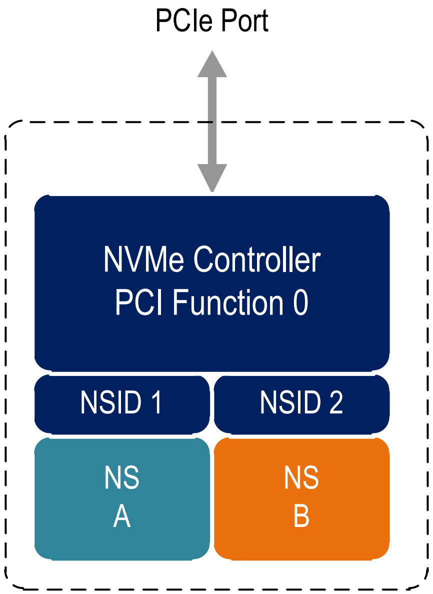
NVMe Subsystem Example

Single controller, single namespace



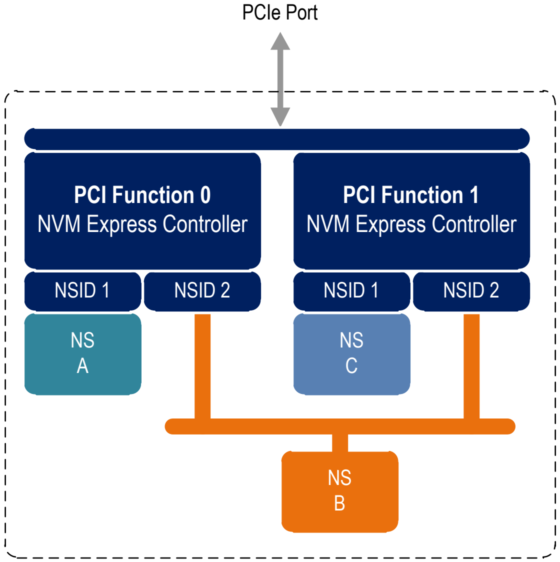
- NS = Namespace, amount of NVM storage formatted for block access
- NSID = Namespace ID, controller unique identifier for namespace (NS)

NVM Subsystem Example
Single Controller, multiple Namespaces

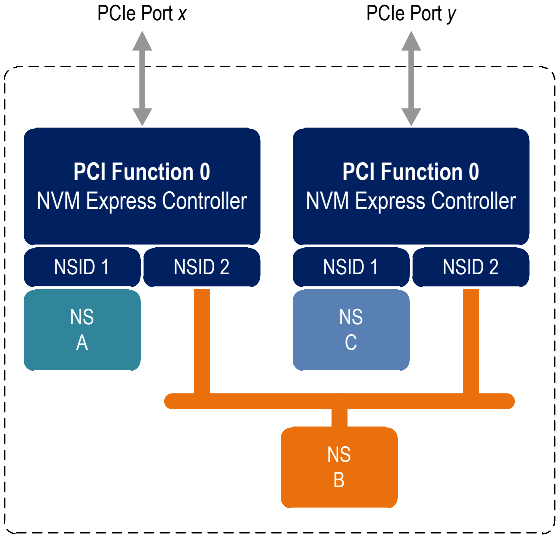


- NS = Namespace, amount of NVM storage formatted for block access
- NSID = Namespace ID, controller unique identifier for namespace (NS)

NVM Subsystem Example
Multiple controllers

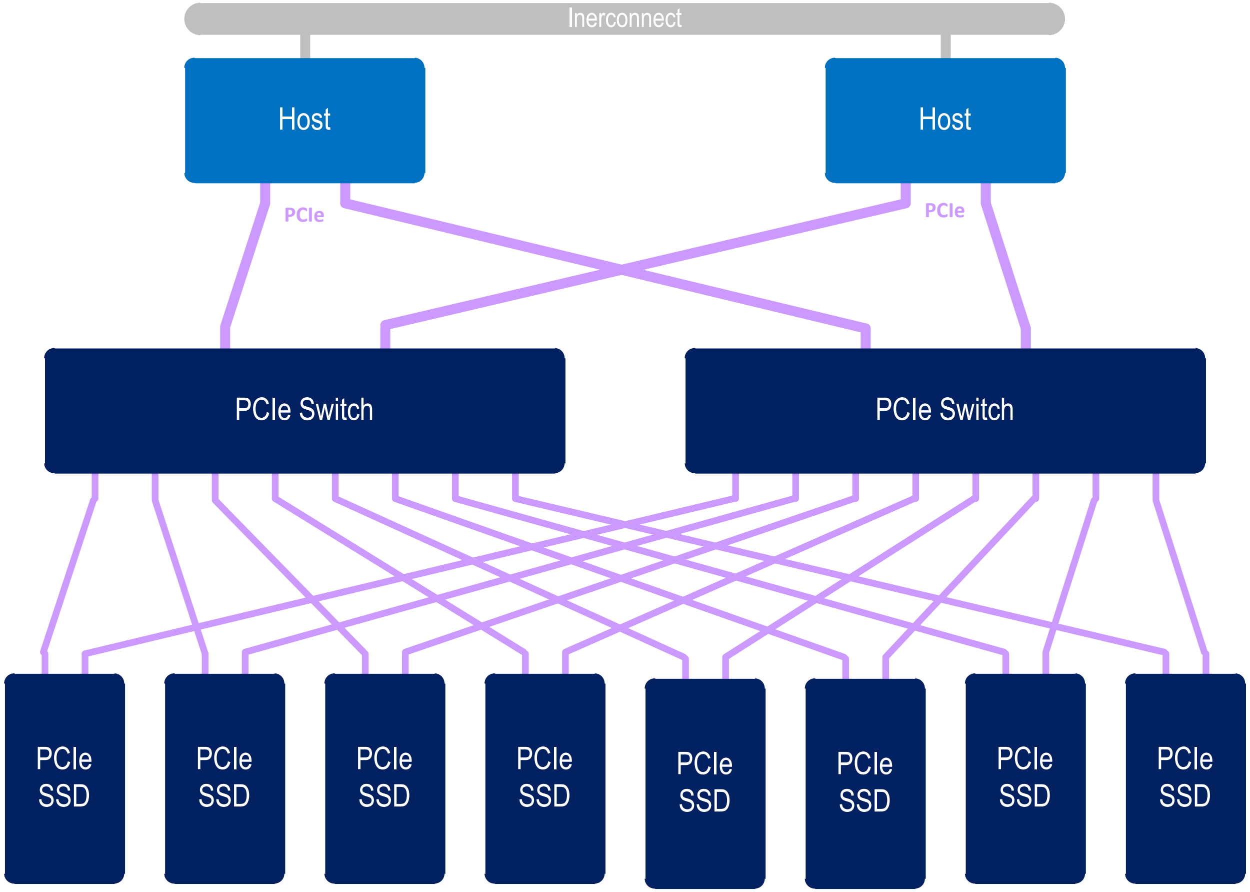


NVM Subsystem with Two Controllers and One Port



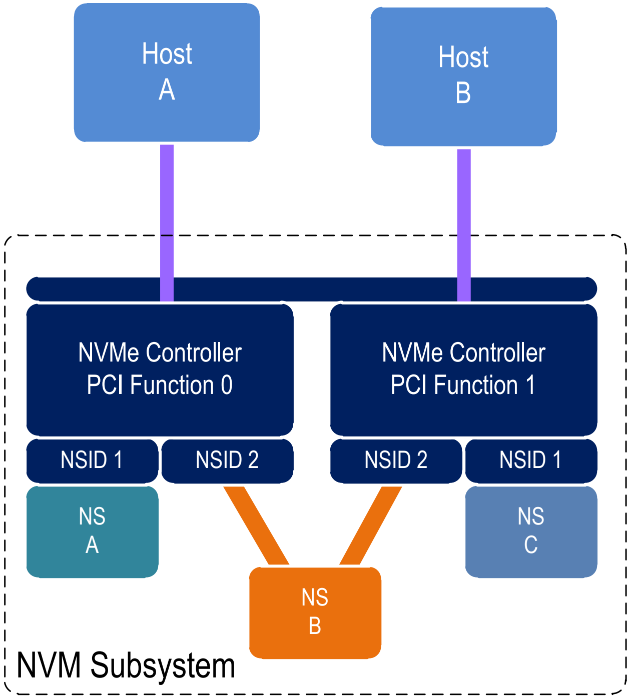
NVM Subsystem with Two Controllers and Two Ports

PCIe Multi-Path Usage Model



Uniquely Identifying a Namespace

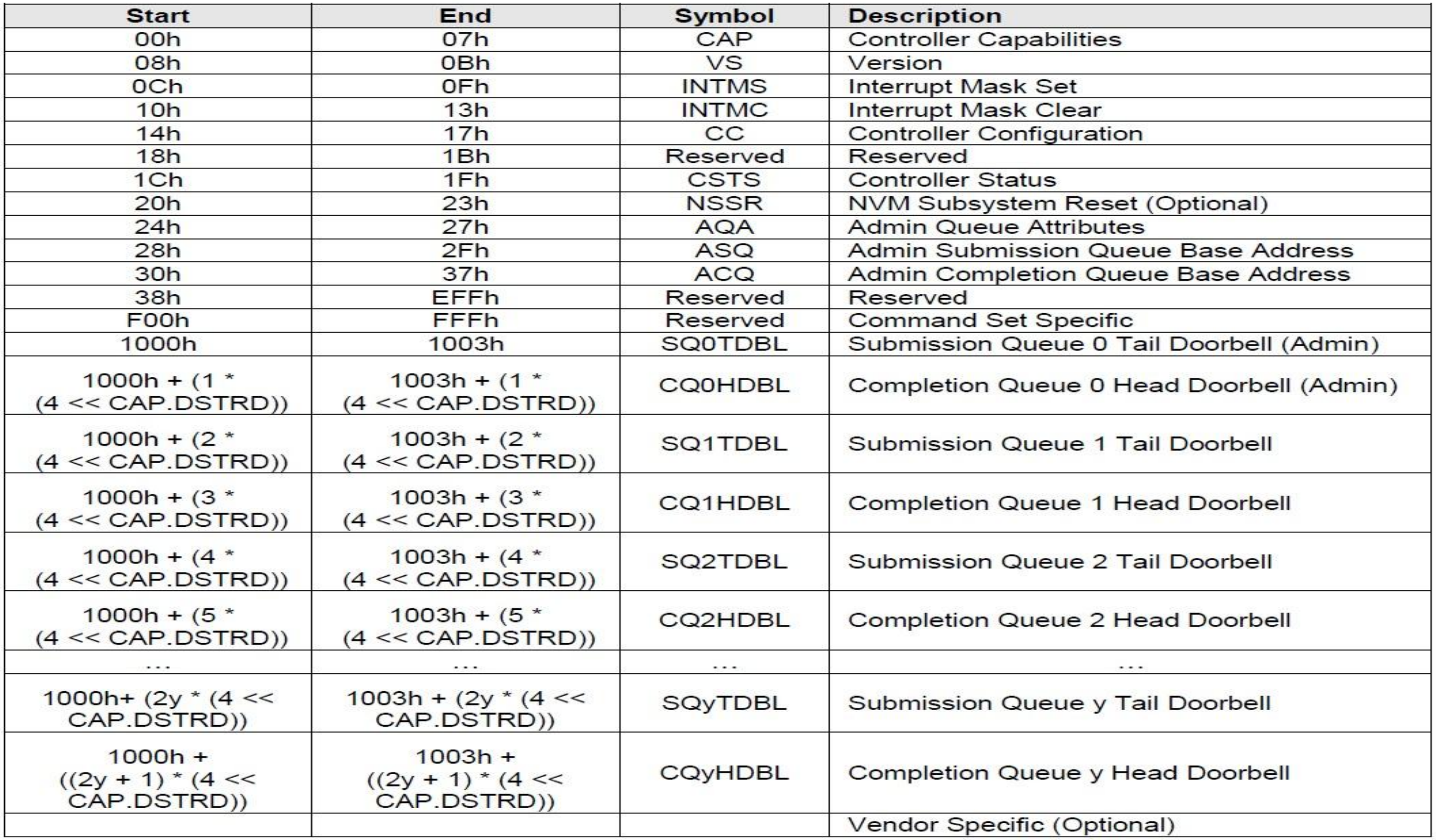
- How do Host A and Host B know that NS B is the same namespace?
- NVM Express 1.1 added unique identifiers for:
  - The NVMe Controller; and
  - Each Namespace within an NVM Subsystem
- These identifiers are guaranteed to be globally unique



Unique NVMe Controller Identifier (64B) =
2B PCI Vendor ID + 20B Serial Number + 40B Model Number + 2B Controller ID

Unique Namespace Identifier (8B) = *8B IEEE Extended Unique Identifier*

NVMe controller register map



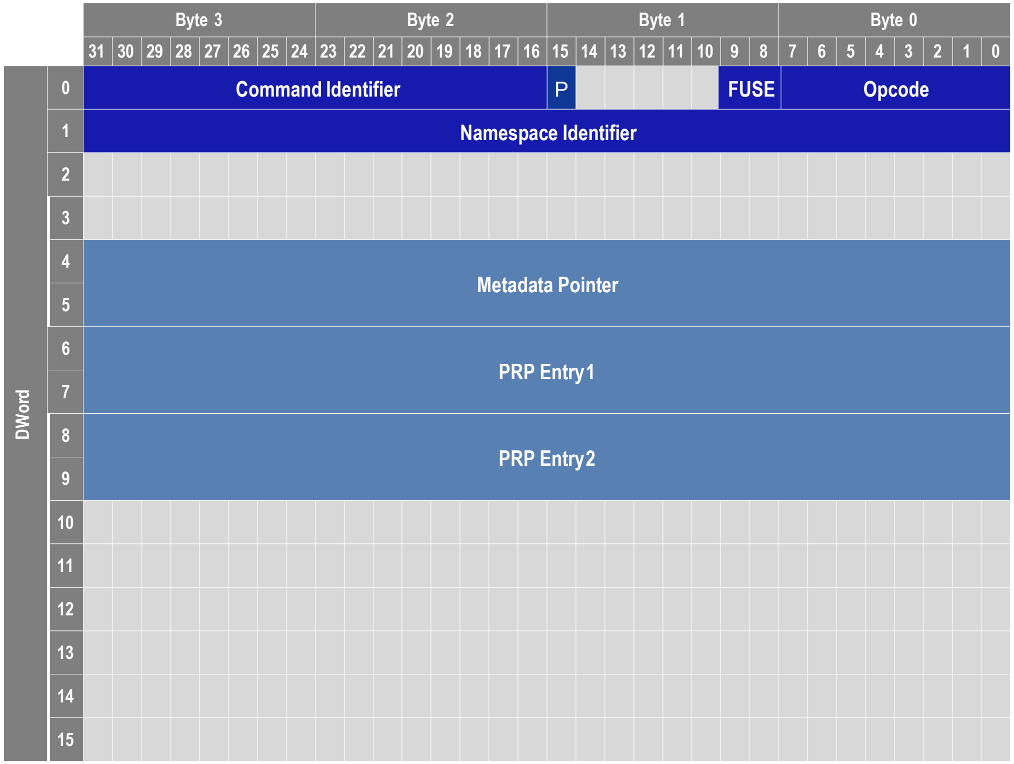
Start	End	Symbol	Description
00h	07h	CAP	Controller Capabilities
08h	0Bh	VS	Version
0Ch	0Fh	INTMS	Interrupt Mask Set
10h	13h	INTMC	Interrupt Mask Clear
14h	17h	CC	Controller Configuration
18h	1Bh	Reserved	Reserved
1Ch	1Fh	CSTS	Controller Status
20h	23h	NSSR	NVM Subsystem Reset (Optional)
24h	27h	AQA	Admin Queue Attributes
28h	2Fh	ASQ	Admin Submission Queue Base Address
30h	37h	ACQ	Admin Completion Queue Base Address
38h	EFFh	Reserved	Reserved
F00h	FFFh	Reserved	Command Set Specific
1000h	1003h	SQ0TDBL	Submission Queue 0 Tail Doorbell (Admin)
1000h + (1 * (4 << CAP.DSTRD))	1003h + (1 * (4 << CAP.DSTRD))	CQ0HDBL	Completion Queue 0 Head Doorbell (Admin)
1000h + (2 * (4 << CAP.DSTRD))	1003h + (2 * (4 << CAP.DSTRD))	SQ1TDBL	Submission Queue 1 Tail Doorbell
1000h + (3 * (4 << CAP.DSTRD))	1003h + (3 * (4 << CAP.DSTRD))	CQ1HDBL	Completion Queue 1 Head Doorbell
1000h + (4 * (4 << CAP.DSTRD))	1003h + (4 * (4 << CAP.DSTRD))	SQ2TDBL	Submission Queue 2 Tail Doorbell
1000h + (5 * (4 << CAP.DSTRD))	1003h + (5 * (4 << CAP.DSTRD))	CQ2HDBL	Completion Queue 2 Head Doorbell
…	…	…	…
1000h+ (2y * (4 << CAP.DSTRD))	1003h + (2y * (4 << CAP.DSTRD))	SQyTDBL	Submission Queue y Tail Doorbell
1000h + ((2y + 1) * (4 << CAP.DSTRD))	1003h + ((2y + 1) * (4 << CAP.DSTRD))	CQyHDBL	Completion Queue y Head Doorbell
			Vendor Specific (Optional)

Controller Initialization

The host performs the following actions in sequence to initialize the controller to begin executing Admin commands:

1. Set the PCI and PCI Express registers based on the system configuration. This includes configuration of power management features. Pin-based or single-message MSI interrupts should be used until the number of I/O Queues is determined.
2. Configure the Admin Queue by setting the Admin Queue Attributes (AQA), Admin Submission Queue Base Address (ASQ), and Admin Completion Queue Base Address (ACQ) to appropriate values.
3. Configure:
   1. the arbitration mechanism in CC.AMS
   2. the memory page size in CC.MPS
   3. the I/O Command Set in CC.CSS
4. Enable the controller by setting CC.EN to ‘1’.
5. Wait for the controller to indicate it is ready to process commands (i.e., when CSTS.RDY is set to ‘1’)

Submission Queue Element with PRPs(64B)

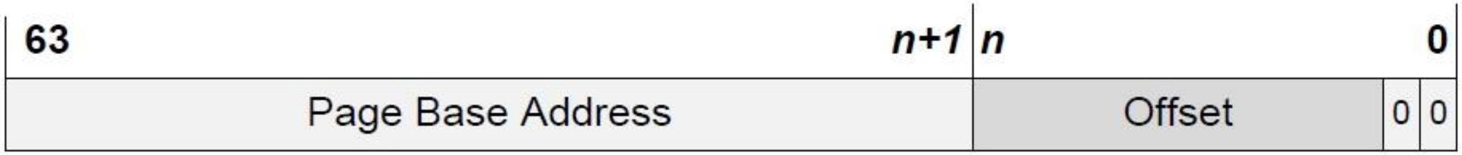


DWord	Byte 3	Byte 2	Byte 1			Byte 0
	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15	14 13 12 11 10	9 8	7 6 5 4 3 2 1 0
0	Command Identifier		P		FUSE	Opcode
1	Namespace Identifier					
2						
3						
4	Metadata Pointer					
5						
6	PRP Entry1					
7						
8	PRP Entry2					
9						
10						
11						
12						
13						
14						
15						

- **Opcode** – Command operation code
- **Fused Operation (FUSE)** – specifies if two commands should be executed as atomic unit (optional)
- **PRP or SGL for Data Transfer** = 0 specifies that PRP's are used; 1 specifies SGLs are used
- **Command Identifier** – Command ID within submission queue
- **Namespace** – Namespace on which command operates
- **Metadata Pointer** – Pointer to contiguous buffer containing metadata
- **PRP Entry 1** – First PRP entry for the command or PRP list pointer depending on the command
- **PRP Entry 2** – Second PRP entry for the command or PRP list pointer depending on the command

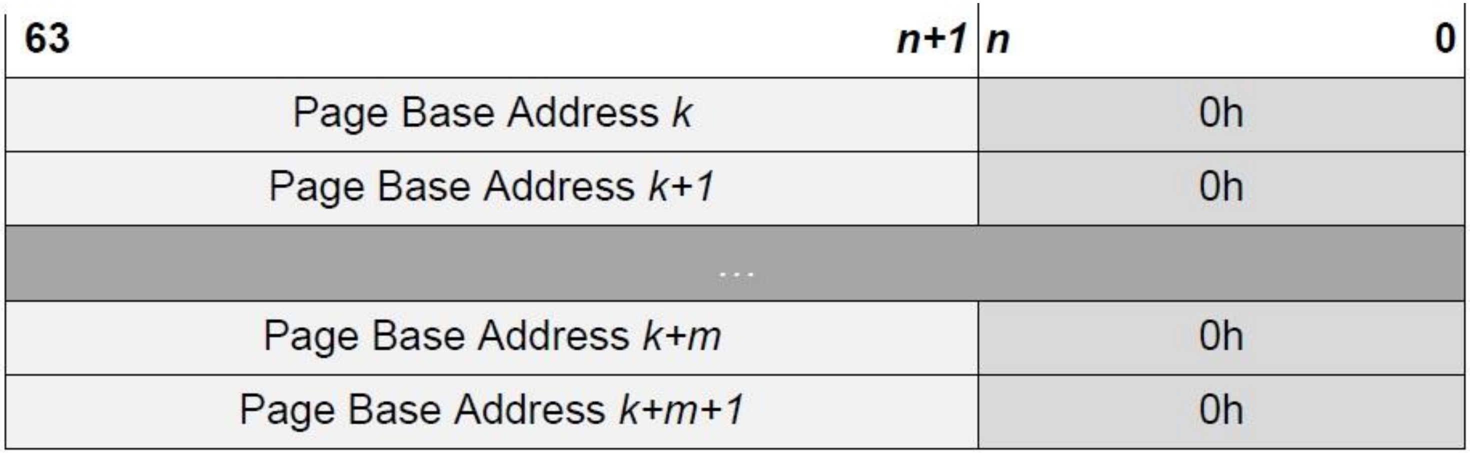
Physical Region Pages (PRPs)

- PRP contains the 64-bit physical memory page address. The lower bits (n:2) of this field indicate the offset within the memory page. N is defined by the memory page size (CC.MPS)



63 *n+1*	*n* 0		
Page Base Address	Offset	0	0

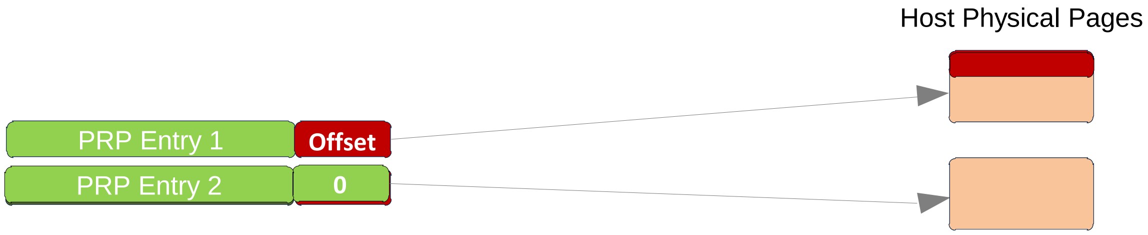
- PRP List contains a list of PRPs with generally no offsets.



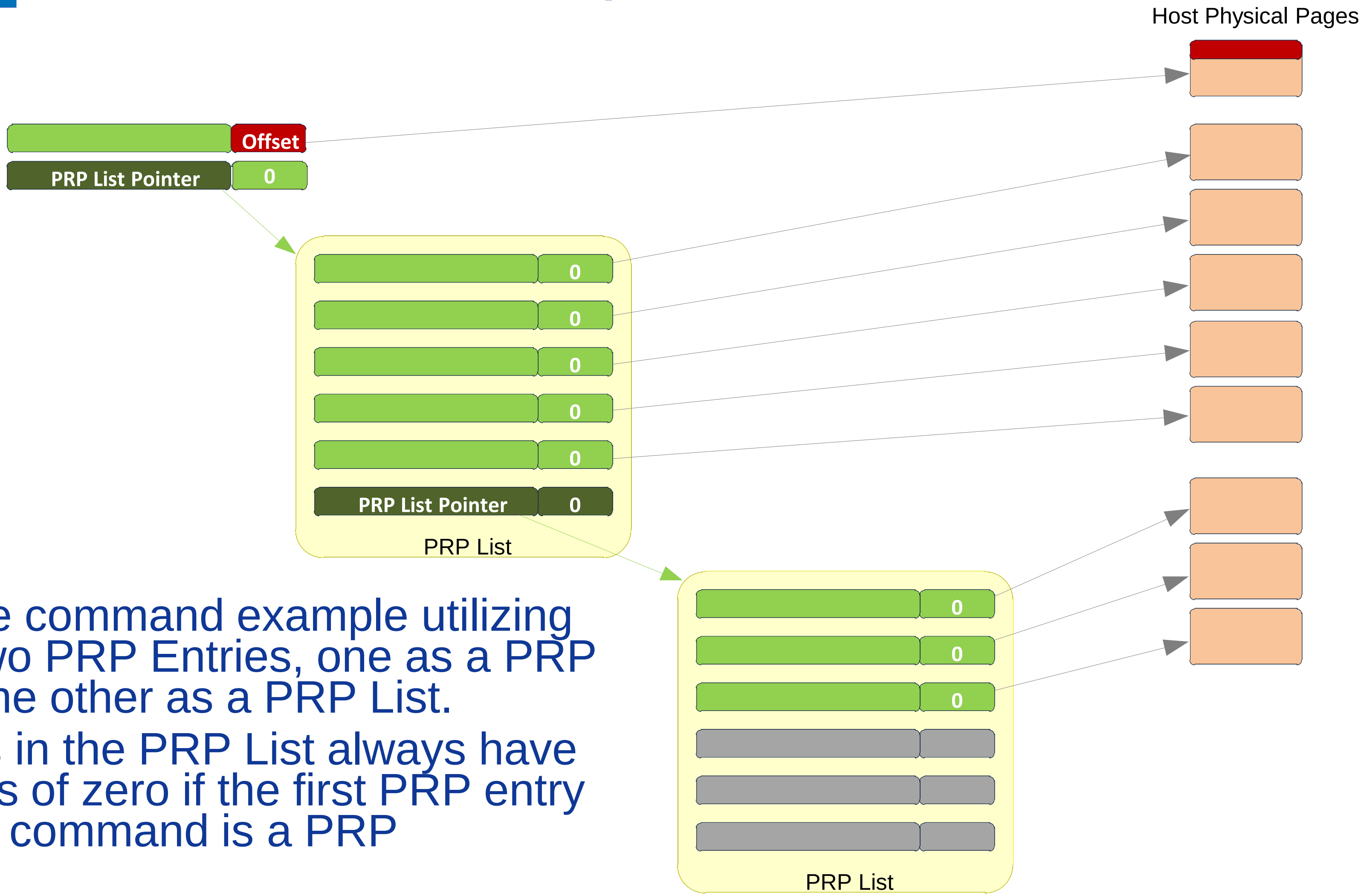
63 *n+1*	*n* 0
Page Base Address *k*	0h
Page Base Address *k+1*	0h
...	
Page Base Address *k+m*	0h
Page Base Address *k+m+1*	0h

PRP Example

- NVMe command example utilizing the two PRP Entries as PRPs. The first PRP has an offset into the memory page.

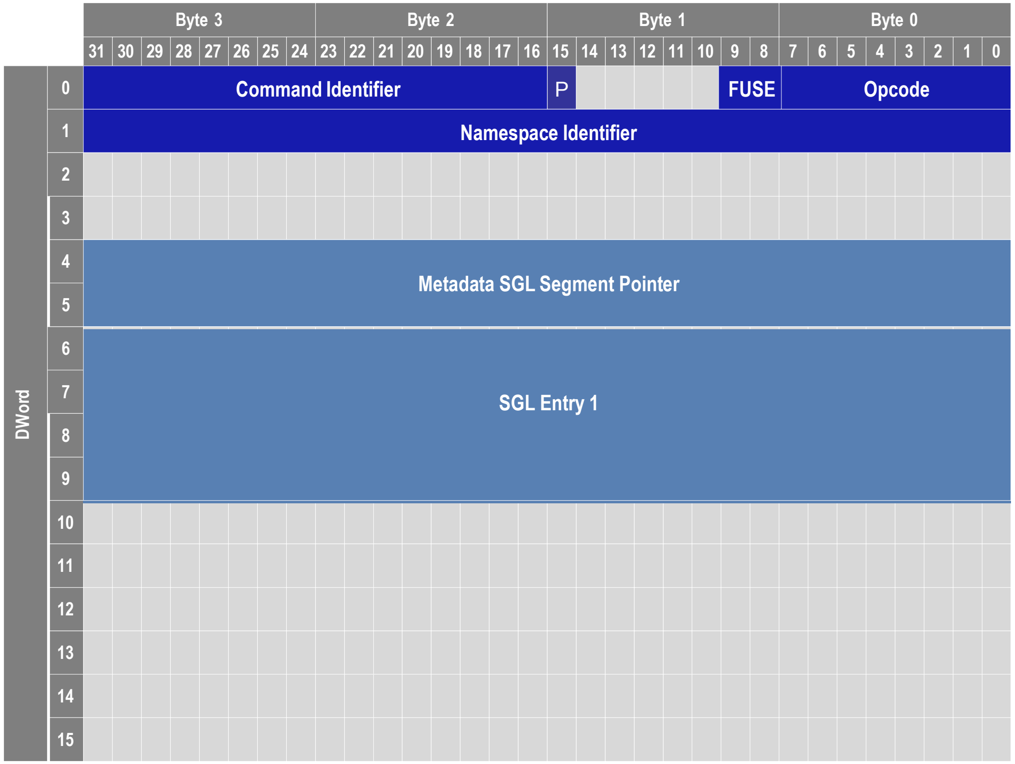


PRP List Example



- NVMe command example utilizing the two PRP Entries, one as a PRP and the other as a PRP List.
- PRPs in the PRP List always have offsets of zero if the first PRP entry in the command is a PRP

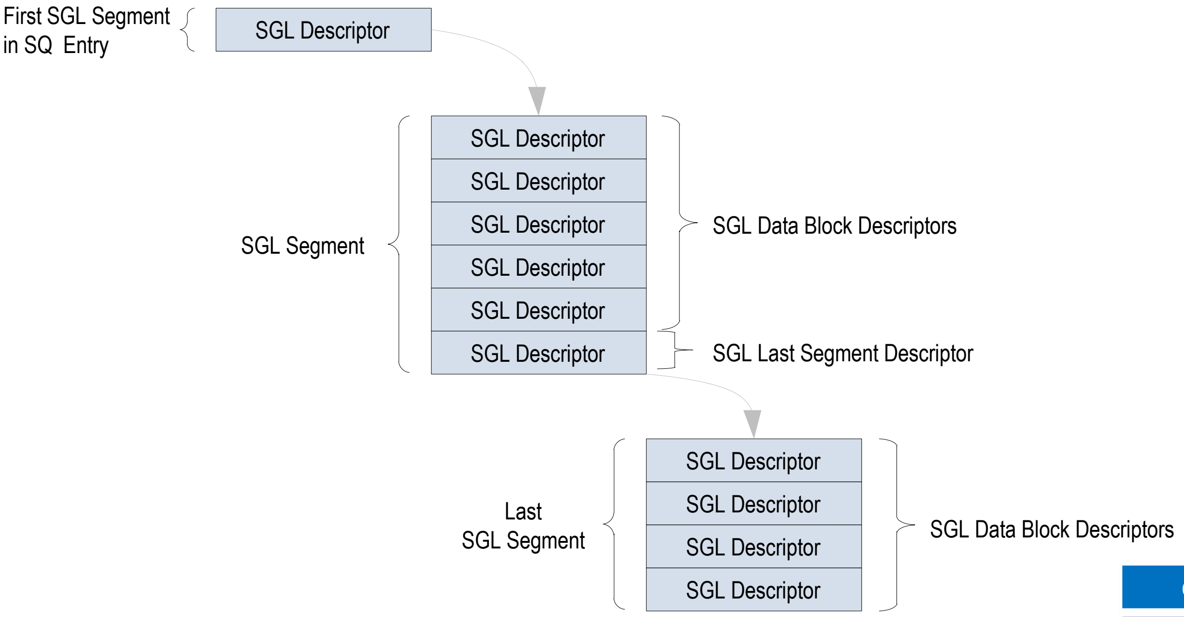
Submission Queue Element with SGLs(64B)



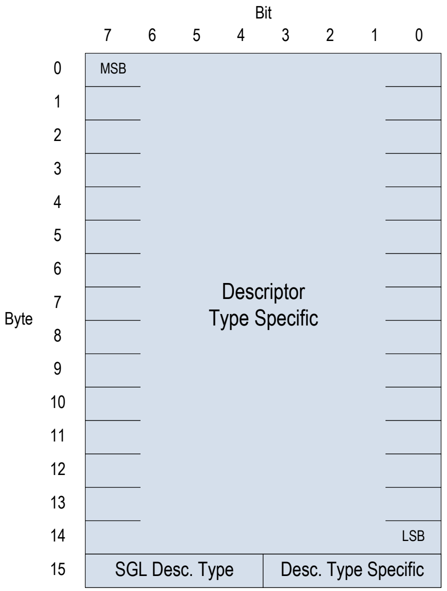
- **Opcode** – Command operation code
- **Fused Operation (FUSE)** – specifies if two commands should be executed as atomic unit (optional)
- **PRP or SGL for Data Transfer** = 0 specifies that PRP's are used ; 1 specifies that SGLs are used
- **Command Identifier** – Command ID within submission queue
- **Namespace** – Namespace on which command operates
- **Metadata SQL Segment Pointer** – first SGL segment which describes the metadata to transfer
- **SGL Entry 1** – the first SGL segment for the command

Scatter Gather List (SGL)

SGL List

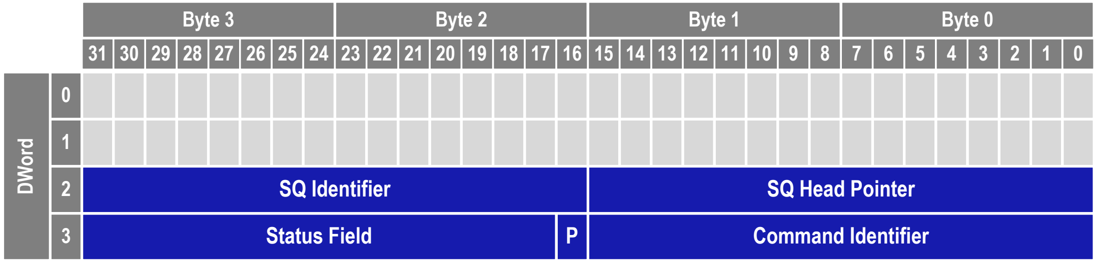


SGL Descriptor



Code	Descriptor Type
0h	SGL Data Block
1h	SGL Bit Bucket
2h	SGL Segment
3h	SGL Last Segment
4h - Eh	Reserved
Fh	Vendor Specific

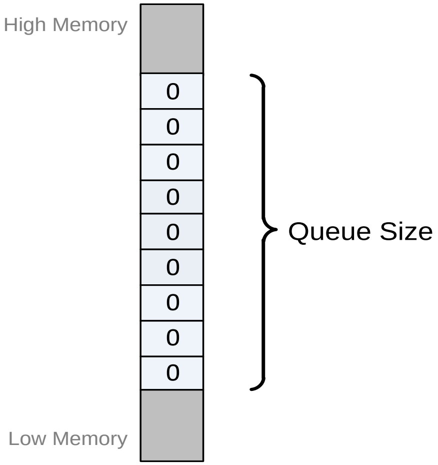
Completion Queue Element (16B)



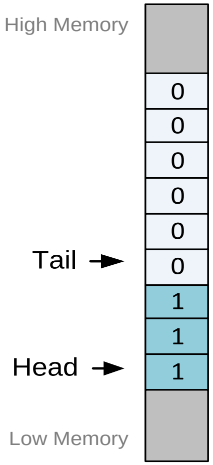
		Byte 3								Byte 2								Byte 1								Byte 0							
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DWord	0																																
	1																																
	2	SQ Identifier																SQ Head Pointer															
	3	Status Field															P	Command Identifier															

- **SQ Head Pointer** – Submission queue head pointer associated with SQ Identifier
- **SQ Identifier** – Submission queue associated with completed command
- **Command Identifier** – Command ID within submission queue
- **Phase Tag (P)** – Indicates when new command is reached
- **Status Field** – Status associated with completed command
    - A value of zero indicates successful command completion

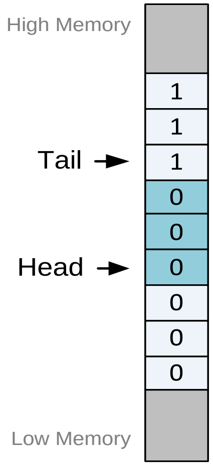
Phase Tag


High Memory
0
0
0
0
0
0
0
0
0
Queue Size
Low Memory

Completion Queue Initial State


High Memory
0
0
0
0
0
Tail → 0
1
1
Head → 1
Low Memory

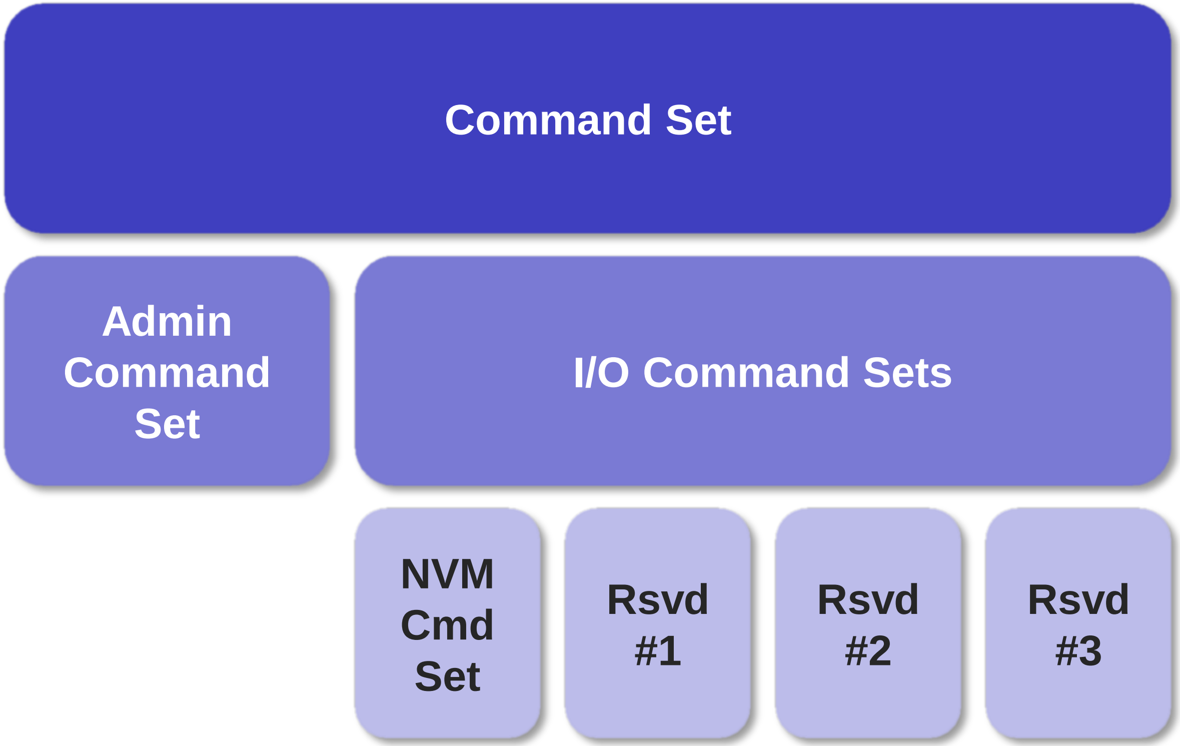
Invert Phase Tag for Each Completion Entry Write (Odd Pass 1,3,5 …)


High Memory
1
1
Tail → 1
0
0
Head → 0
0
0
0
Low Memory

Invert Phase Tag for Each Completion Entry Write (Even Pass 2,4,6 …)

- Phase Tag Operation
  - Initially zero
  - Controller “inverts” phase tag of an entry each time it writes a completion entry
  - Host knows phase tag of completions and can determine when last full entry is reached

NVMe Command Sets



Command Set

Admin Command Set

I/O Command Sets

NVM Cmd Set

Rsvd #1

Rsvd #2

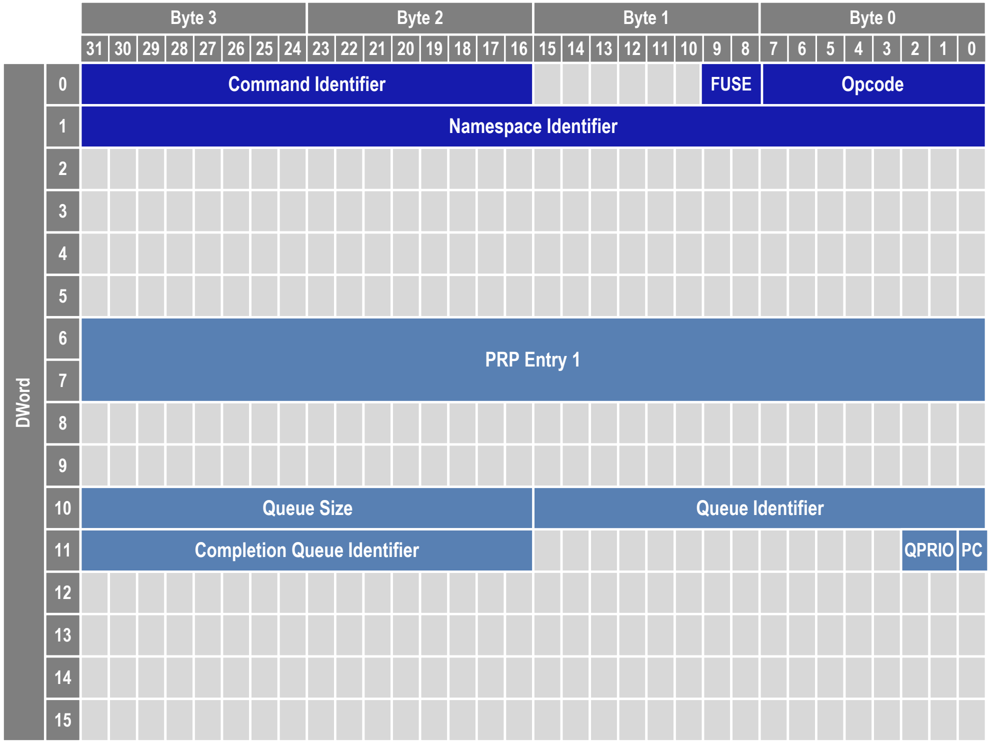
Rsvd #3

Admin Commands

Command	Required or Optional	Category
Create I/O Submission Queue	Required	Queue Management
Delete I/O Submission Queue	Required	
Create I/O Completion Queue	Required	
Delete I/O Completion Queue	Required	
Identify	Required	Configuration
Get Features	Required	
Set Features	Required	
Get Log Page	Required	Status Reporting
Asynchronous Event Request	Required	
Abort	Required	Abort Command
Firmware Image Download	Optional	Firmware Update / Management
Firmware Activate	Optional	
I/O Command Set Specific Commands	Optional	I/O Command Set Specific
Vendor Specific Commands	Optional	Vendor Specific

All Admin command use PRPs

Create I/O Submission Queue

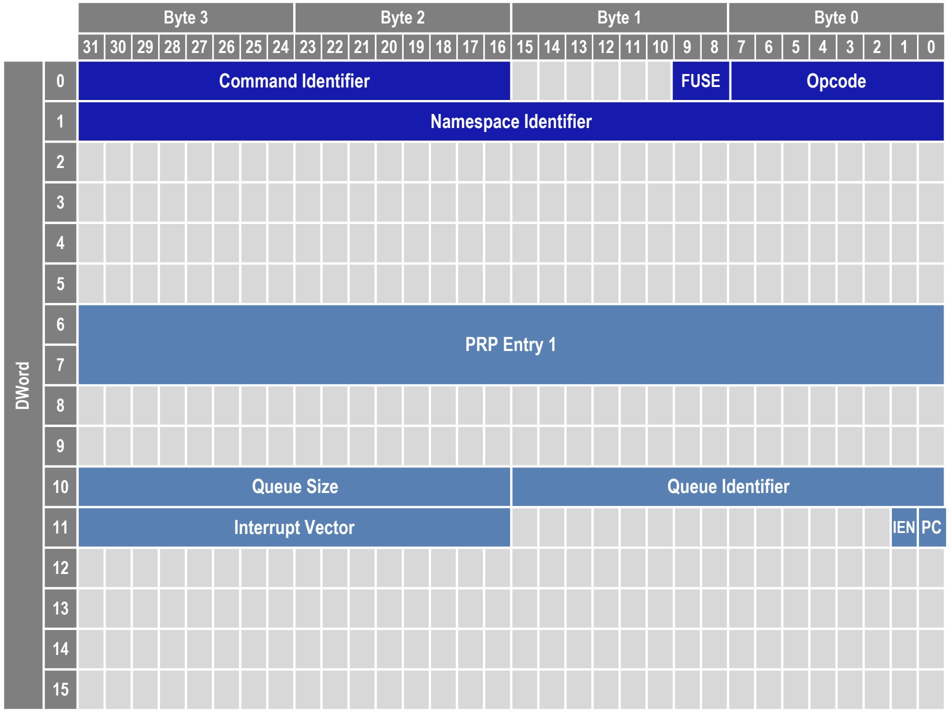


<table>
<tr><td rowspan="2"></td><td rowspan="2"></td><td colspan="8">Byte 3</td><td colspan="8">Byte 2</td><td colspan="8">Byte 1</td><td colspan="8">Byte 0</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td rowspan="16">DWord</td><td>0</td><td colspan="16">Command Identifier</td><td colspan="6"></td><td colspan="2">FUSE</td><td colspan="8">Opcode</td></tr>
<tr><td>1</td><td colspan="32">Namespace Identifier</td></tr>
<tr><td>2</td><td colspan="32"></td></tr>
<tr><td>3</td><td colspan="32"></td></tr>
<tr><td>4</td><td colspan="32"></td></tr>
<tr><td>5</td><td colspan="32"></td></tr>
<tr><td>6</td><td colspan="32" rowspan="2">PRP Entry 1</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td><td colspan="32"></td></tr>
<tr><td>9</td><td colspan="32"></td></tr>
<tr><td>10</td><td colspan="16">Queue Size</td><td colspan="16">Queue Identifier</td></tr>
<tr><td>11</td><td colspan="16">Completion Queue Identifier</td><td colspan="14"></td><td colspan="2">QPRIO</td><td>PC</td></tr>
<tr><td>12</td><td colspan="32"></td></tr>
<tr><td>13</td><td colspan="32"></td></tr>
<tr><td>14</td><td colspan="32"></td></tr>
<tr><td>15</td><td colspan="32"></td></tr>
</table>

<u>Create specified I/O submission queue</u>

- **Queue Identifier** – Submission queue ID number
- **Queue Size** – Number of entries in submission queue (zero based value)
- **Completion Queue Identifier** – Completion queue ID number associated with submission queue
- **Queue Priority (QPRIO)** – Queue Priority when WRR with urgent priority service class priority is selected
- **Physically Contiguous (PC)**
  - 1- Submission queue is physically contiguous in host memory
  - 0 – Submission queue is not physically contiguous
- **PRP Entry 1** – When not physically contiguous, this is a pointer to a PRP list that contains host pages
- **Command Specific Error Values**
  - Completion Queue Invalid
  - Invalid Queue Identifier
  - Maximum Queue Size Exceeded

Create I/O Completion Queue

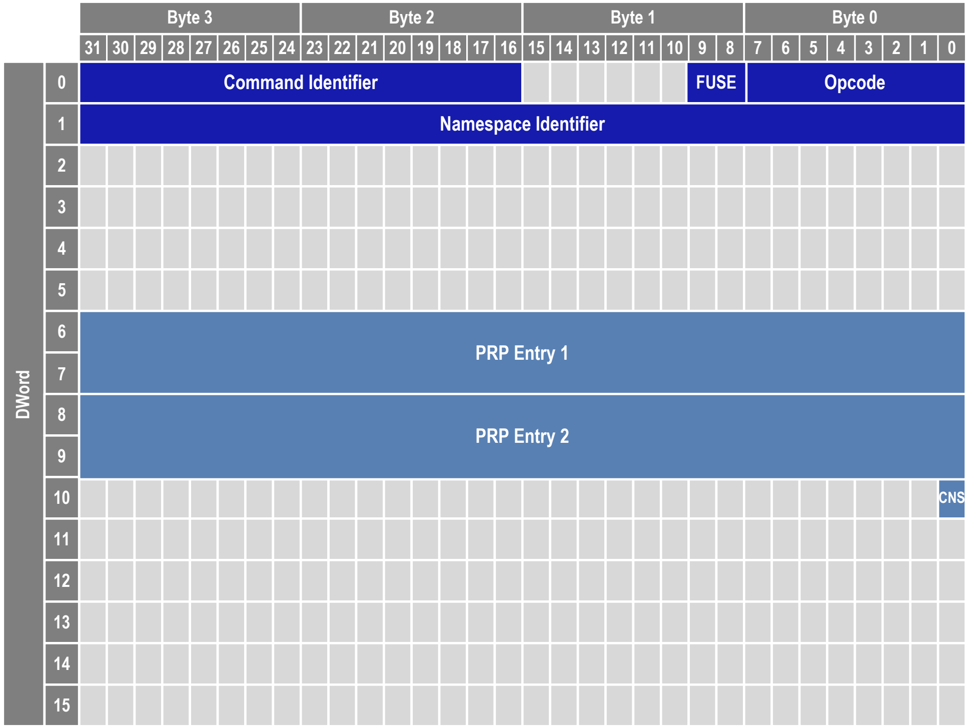


<table>
<tr><th></th><th></th><th colspan="8">Byte 3</th><th colspan="8">Byte 2</th><th colspan="8">Byte 1</th><th colspan="8">Byte 0</th></tr>
<tr><th></th><th></th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="16">DWord</td><td>0</td><td colspan="16">Command Identifier</td><td colspan="6"></td><td colspan="2">FUSE</td><td colspan="8">Opcode</td></tr>
<tr><td>1</td><td colspan="32">Namespace Identifier</td></tr>
<tr><td>2</td><td colspan="32"></td></tr>
<tr><td>3</td><td colspan="32"></td></tr>
<tr><td>4</td><td colspan="32"></td></tr>
<tr><td>5</td><td colspan="32"></td></tr>
<tr><td>6</td><td colspan="32" rowspan="2">PRP Entry 1</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td><td colspan="32"></td></tr>
<tr><td>9</td><td colspan="32"></td></tr>
<tr><td>10</td><td colspan="16">Queue Size</td><td colspan="16">Queue Identifier</td></tr>
<tr><td>11</td><td colspan="16">Interrupt Vector</td><td colspan="14"></td><td>IEN</td><td>PC</td></tr>
<tr><td>12</td><td colspan="32"></td></tr>
<tr><td>13</td><td colspan="32"></td></tr>
<tr><td>14</td><td colspan="32"></td></tr>
<tr><td>15</td><td colspan="32"></td></tr>
</table>

Create specified I/O completion queue

- **Queue Identifier** – completion queue ID number
- **Queue Size** – Number of entries in completion queue (zero based value)
- **Interrupt Vector** – MSI-X or MSI vector number
- **Interrupt Enable (IEN)**
  - 0 – Interrupts disabled
  - 1 – Interrupts enabled
- **Physically Contiguous (PC)**
  - 1- Submission queue is physically contiguous in host memory
  - 0 – Submission queue is not physically contiguous
- **PRP Entry 1** – When not physically contiguous, this is a pointer to a PRP list that contains host pages

Identify



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

DWord

0 Command Identifier | FUSE | Opcode

1 Namespace Identifier

2 3 4 5

6 7 PRP Entry 1

8 9 PRP Entry 2

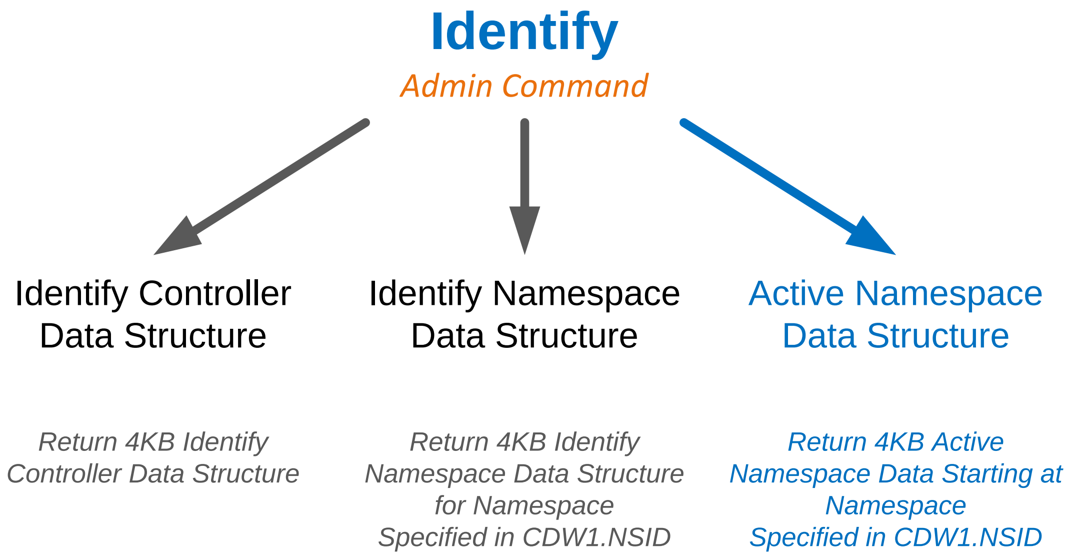
10 CNS

11 12 13 14 15

Returns up to 4KB data structure that describes controller or namespace

- **PRP Entry 1** – Starting address of where 4KB data structure is to be written
  - Offset may be non-zero
- **PRP Entry 2** – Starting address of where remainder of 4KB data structure is to be written
- **Controller or Namespace Structure (CNS)**
  - 00b – Return corresponding namespace data structure
  - 01b – Return corresponding controller data structure
  - 10b – Return list of 1024 active namespace IDs starting at the Namespace Identifer.

Active Namespace Reporting



Identify

Admin Command

Identify Controller Data Structure

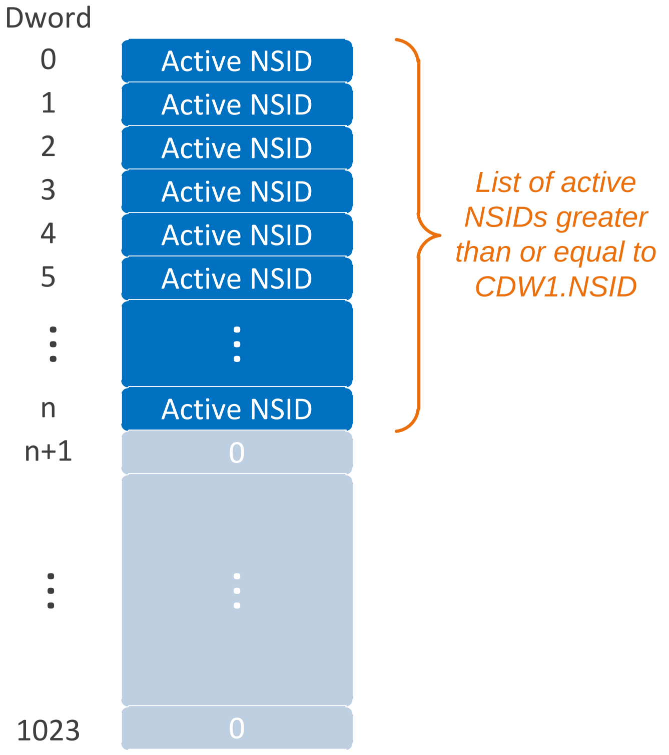
Return 4KB Identify Controller Data Structure

Identify Namespace Data Structure

Return 4KB Identify Namespace Data Structure for Namespace Specified in CDW1.NSID

Active Namespace Data Structure

Return 4KB Active Namespace Data Starting at Namespace Specified in CDW1.NSID

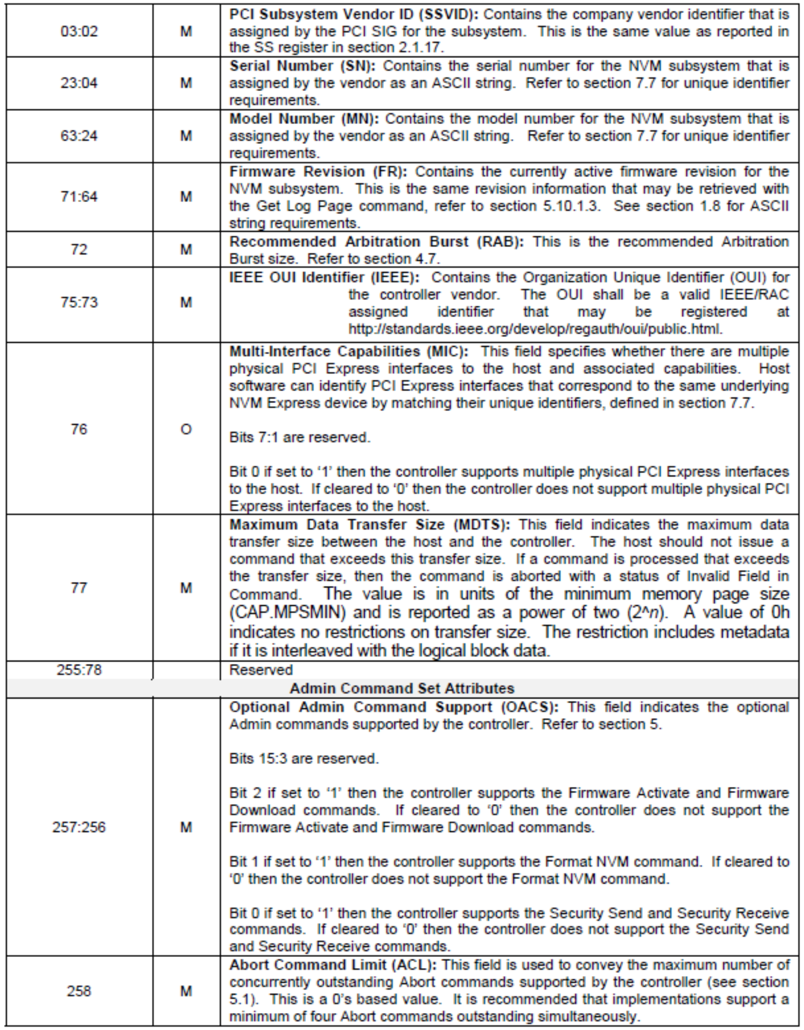


Dword	
0	Active NSID
1	Active NSID
2	Active NSID
3	Active NSID
4	Active NSID
5	Active NSID
⋮	⋮
n	Active NSID
n+1	0
⋮	⋮
1023	0

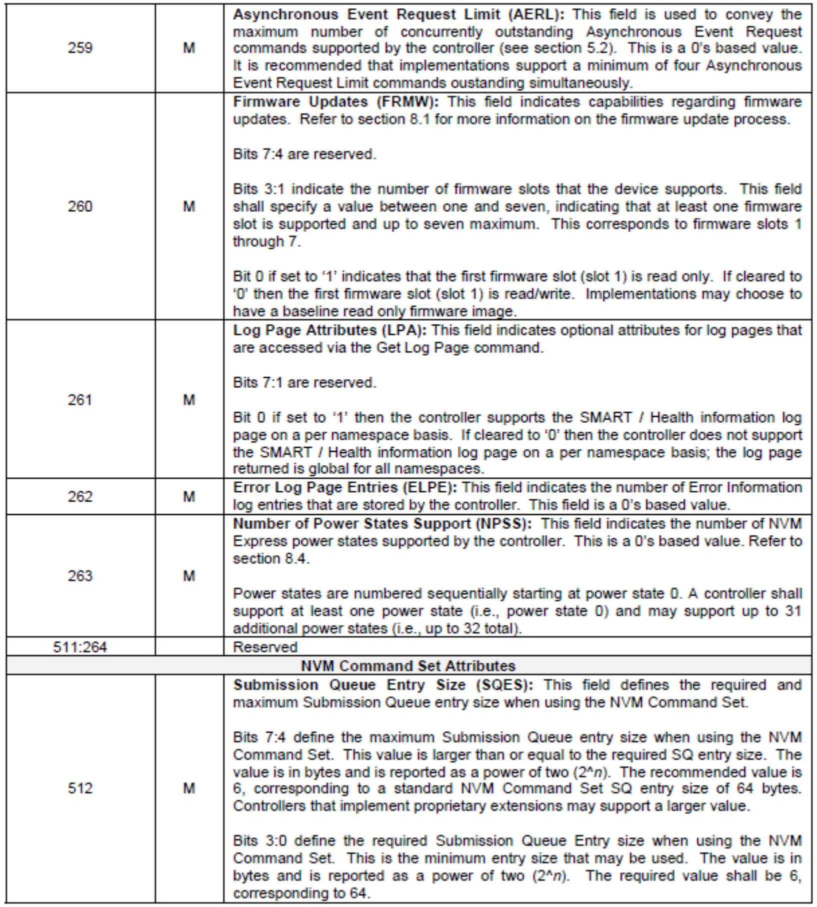
List of active NSIDs greater than or equal to CDW1.NSID

Active Namespace Data Structure

Identify Controller Data Structure



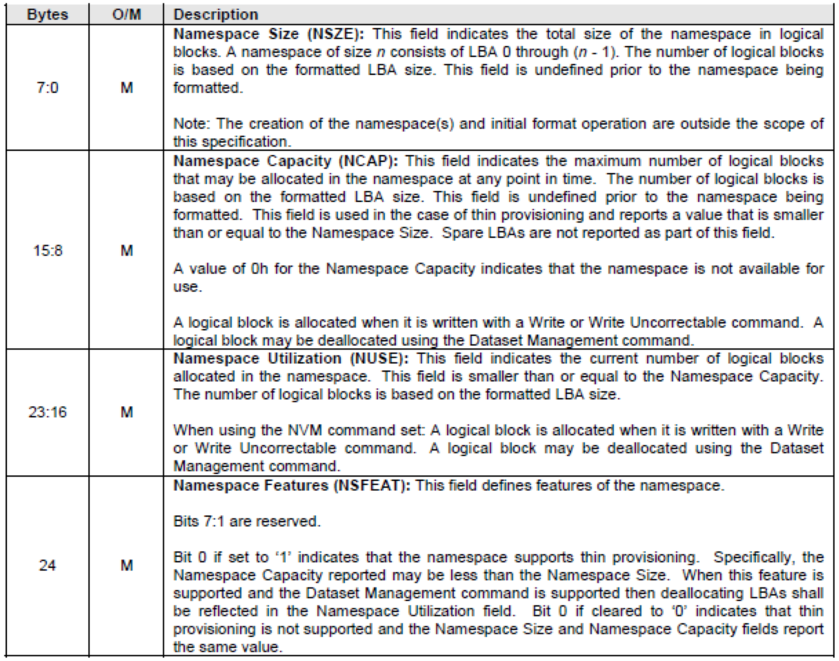
03:02	M	**PCI Subsystem Vendor ID (SSVID):** Contains the company vendor identifier that is assigned by the PCI SIG for the subsystem. This is the same value as reported in the SS register in section 2.1.17.
23:04	M	**Serial Number (SN):** Contains the serial number for the NVM subsystem that is assigned by the vendor as an ASCII string. Refer to section 7.7 for unique identifier requirements.
63:24	M	**Model Number (MN):** Contains the model number for the NVM subsystem that is assigned by the vendor as an ASCII string. Refer to section 7.7 for unique identifier requirements.
71:64	M	**Firmware Revision (FR):** Contains the currently active firmware revision for the NVM subsystem. This is the same revision information that may be retrieved with the Get Log Page command, refer to section 5.10.1.3. See section 1.8 for ASCII string requirements.
72	M	**Recommended Arbitration Burst (RAB):** This is the recommended Arbitration Burst size. Refer to section 4.7.
75:73	M	**IEEE OUI Identifier (IEEE):** Contains the Organization Unique Identifier (OUI) for the controller vendor. The OUI shall be a valid IEEE/RAC assigned identifier that may be registered at http://standards.ieee.org/develop/regauth/oui/public.html.
76	O	**Multi-Interface Capabilities (MIC):** This field specifies whether there are multiple physical PCI Express interfaces to the host and associated capabilities. Host software can identify PCI Express interfaces that correspond to the same underlying NVM Express device by matching their unique identifiers, defined in section 7.7. Bits 7:1 are reserved. Bit 0 if set to '1' then the controller supports multiple physical PCI Express interfaces to the host. If cleared to '0' then the controller does not support multiple physical PCI Express interfaces to the host.
77	M	**Maximum Data Transfer Size (MDTS):** This field indicates the maximum data transfer size between the host and the controller. The host should not issue a command that exceeds this transfer size. If a command is processed that exceeds the transfer size, then the command is aborted with a status of Invalid Field in Command. The value is in units of the minimum memory page size (CAP.MPSMIN) and is reported as a power of two (2^*n*). A value of 0h indicates no restrictions on transfer size. The restriction includes metadata if it is interleaved with the logical block data.
255:78		Reserved
		Admin Command Set Attributes
257:256	M	**Optional Admin Command Support (OACS):** This field indicates the optional Admin commands supported by the controller. Refer to section 5. Bits 15:3 are reserved. Bit 2 if set to '1' then the controller supports the Firmware Activate and Firmware Download commands. If cleared to '0' then the controller does not support the Firmware Activate and Firmware Download commands. Bit 1 if set to '1' then the controller supports the Format NVM command. If cleared to '0' then the controller does not support the Format NVM command. Bit 0 if set to '1' then the controller supports the Security Send and Security Receive commands. If cleared to '0' then the controller does not support the Security Send and Security Receive commands.
258	M	**Abort Command Limit (ACL):** This field is used to convey the maximum number of concurrently outstanding Abort commands supported by the controller (see section 5.1). This is a 0's based value. It is recommended that implementations support a minimum of four Abort commands outstanding simultaneously.



259	M	**Asynchronous Event Request Limit (AERL):** This field is used to convey the maximum number of concurrently outstanding Asynchronous Event Request commands supported by the controller (see section 5.2). This is a 0's based value. It is recommended that implementations support a minimum of four Asynchronous Event Request Limit commands oustanding simultaneously.
260	M	**Firmware Updates (FRMW):** This field indicates capabilities regarding firmware updates. Refer to section 8.1 for more information on the firmware update process. Bits 7:4 are reserved. Bits 3:1 indicate the number of firmware slots that the device supports. This field shall specify a value between one and seven, indicating that at least one firmware slot is supported and up to seven maximum. This corresponds to firmware slots 1 through 7. Bit 0 if set to '1' indicates that the first firmware slot (slot 1) is read only. If cleared to '0' then the first firmware slot (slot 1) is read/write. Implementations may choose to have a baseline read only firmware image.
261	M	**Log Page Attributes (LPA):** This field indicates optional attributes for log pages that are accessed via the Get Log Page command. Bits 7:1 are reserved. Bit 0 if set to '1' then the controller supports the SMART / Health information log page on a per namespace basis. If cleared to '0' then the controller does not support the SMART / Health information log page on a per namespace basis; the log page returned is global for all namespaces.
262	M	**Error Log Page Entries (ELPE):** This field indicates the number of Error Information log entries that are stored by the controller. This field is a 0's based value.
263	M	**Number of Power States Support (NPSS):** This field indicates the number of NVM Express power states supported by the controller. This is a 0's based value. Refer to section 8.4. Power states are numbered sequentially starting at power state 0. A controller shall support at least one power state (i.e., power state 0) and may support up to 31 additional power states (i.e., up to 32 total).
511:264		Reserved
		NVM Command Set Attributes
512	M	**Submission Queue Entry Size (SQES):** This field defines the required and maximum Submission Queue entry size when using the NVM Command Set. Bits 7:4 define the maximum Submission Queue entry size when using the NVM Command Set. This value is larger than or equal to the required SQ entry size. The value is in bytes and is reported as a power of two (2^*n*). The recommended value is 6, corresponding to a standard NVM Command Set SQ entry size of 64 bytes. Controllers that implement proprietary extensions may support a larger value. Bits 3:0 define the required Submission Queue Entry size when using the NVM Command Set. This is the minimum entry size that may be used. The value is in bytes and is reported as a power of two (2^*n*). The required value shall be 6, corresponding to 64.

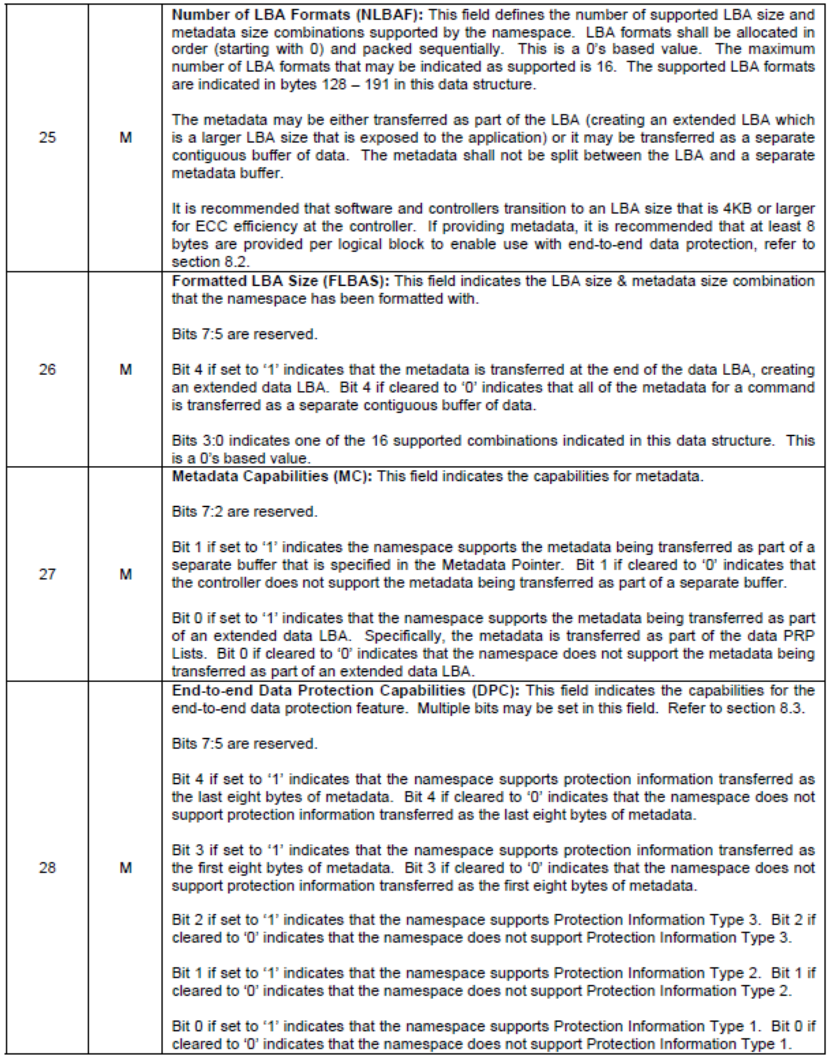
Example Fields – Does Not Show Complete Data Structure

Identify Namespace Data Structure



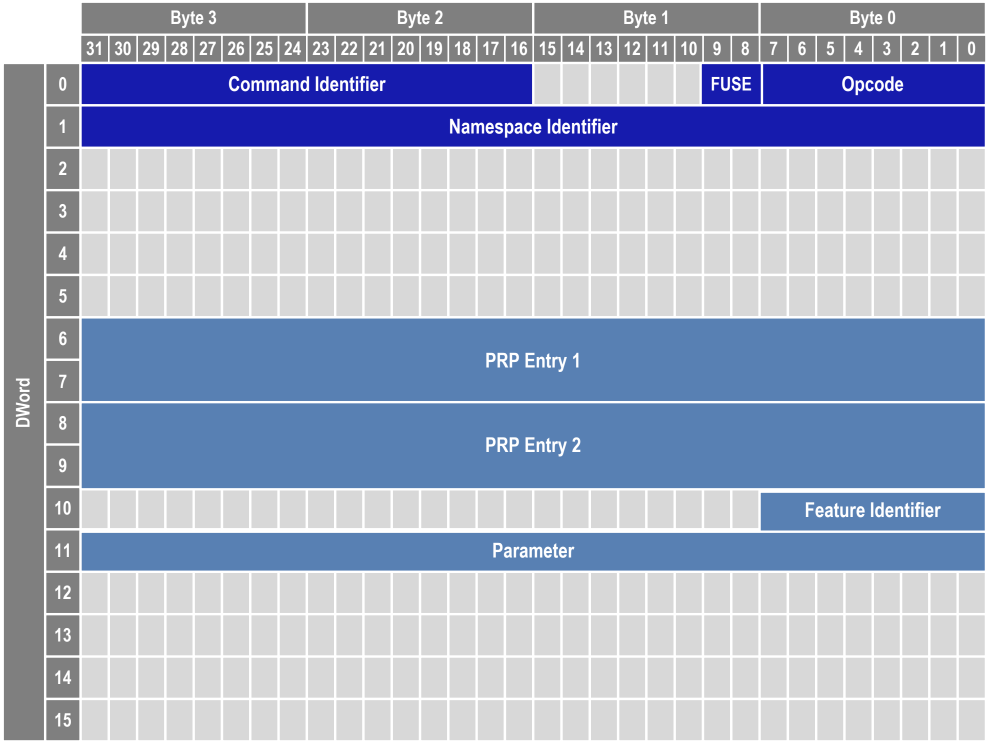
Bytes	O/M	Description
7:0	M	**Namespace Size (NSZE):** This field indicates the total size of the namespace in logical blocks. A namespace of size *n* consists of LBA 0 through (*n* - 1). The number of logical blocks is based on the formatted LBA size. This field is undefined prior to the namespace being formatted. Note: The creation of the namespace(s) and initial format operation are outside the scope of this specification.
15:8	M	**Namespace Capacity (NCAP):** This field indicates the maximum number of logical blocks that may be allocated in the namespace at any point in time. The number of logical blocks is based on the formatted LBA size. This field is undefined prior to the namespace being formatted. This field is used in the case of thin provisioning and reports a value that is smaller than or equal to the Namespace Size. Spare LBAs are not reported as part of this field. A value of 0h for the Namespace Capacity indicates that the namespace is not available for use. A logical block is allocated when it is written with a Write or Write Uncorrectable command. A logical block may be deallocated using the Dataset Management command.
23:16	M	**Namespace Utilization (NUSE):** This field indicates the current number of logical blocks allocated in the namespace. This field is smaller than or equal to the Namespace Capacity. The number of logical blocks is based on the formatted LBA size. When using the NVM command set: A logical block is allocated when it is written with a Write or Write Uncorrectable command. A logical block may be deallocated using the Dataset Management command.
24	M	**Namespace Features (NSFEAT):** This field defines features of the namespace. Bits 7:1 are reserved. Bit 0 if set to '1' indicates that the namespace supports thin provisioning. Specifically, the Namespace Capacity reported may be less than the Namespace Size. When this feature is supported and the Dataset Management command is supported then deallocating LBAs shall be reflected in the Namespace Utilization field. Bit 0 if cleared to '0' indicates that thin provisioning is not supported and the Namespace Size and Namespace Capacity fields report the same value.

Example Fields – Does Not Show Complete Data Structure



25	M	**Number of LBA Formats (NLBAF):** This field defines the number of supported LBA size and metadata size combinations supported by the namespace. LBA formats shall be allocated in order (starting with 0) and packed sequentially. This is a 0's based value. The maximum number of LBA formats that may be indicated as supported is 16. The supported LBA formats are indicated in bytes 128 – 191 in this data structure. The metadata may be either transferred as part of the LBA (creating an extended LBA which is a larger LBA size that is exposed to the application) or it may be transferred as a separate contiguous buffer of data. The metadata shall not be split between the LBA and a separate metadata buffer. It is recommended that software and controllers transition to an LBA size that is 4KB or larger for ECC efficiency at the controller. If providing metadata, it is recommended that at least 8 bytes are provided per logical block to enable use with end-to-end data protection, refer to section 8.2.
26	M	**Formatted LBA Size (FLBAS):** This field indicates the LBA size & metadata size combination that the namespace has been formatted with. Bits 7:5 are reserved. Bit 4 if set to '1' indicates that the metadata is transferred at the end of the data LBA, creating an extended data LBA. Bit 4 if cleared to '0' indicates that all of the metadata for a command is transferred as a separate contiguous buffer of data. Bits 3:0 indicates one of the 16 supported combinations indicated in this data structure. This is a 0's based value.
27	M	**Metadata Capabilities (MC):** This field indicates the capabilities for metadata. Bits 7:2 are reserved. Bit 1 if set to '1' indicates the namespace supports the metadata being transferred as part of a separate buffer that is specified in the Metadata Pointer. Bit 1 if cleared to '0' indicates that the controller does not support the metadata being transferred as part of a separate buffer. Bit 0 if set to '1' indicates that the namespace supports the metadata being transferred as part of an extended data LBA. Specifically, the metadata is transferred as part of the data PRP Lists. Bit 0 if cleared to '0' indicates that the namespace does not support the metadata being transferred as part of an extended data LBA.
28	M	**End-to-end Data Protection Capabilities (DPC):** This field indicates the capabilities for the end-to-end data protection feature. Multiple bits may be set in this field. Refer to section 8.3. Bits 7:5 are reserved. Bit 4 if set to '1' indicates that the namespace supports protection information transferred as the last eight bytes of metadata. Bit 4 if cleared to '0' indicates that the namespace does not support protection information transferred as the last eight bytes of metadata. Bit 3 if set to '1' indicates that the namespace supports protection information transferred as the first eight bytes of metadata. Bit 3 if cleared to '0' indicates that the namespace does not support protection information transferred as the first eight bytes of metadata. Bit 2 if set to '1' indicates that the namespace supports Protection Information Type 3. Bit 2 if cleared to '0' indicates that the namespace does not support Protection Information Type 3. Bit 1 if set to '1' indicates that the namespace supports Protection Information Type 2. Bit 1 if cleared to '0' indicates that the namespace does not support Protection Information Type 2. Bit 0 if set to '1' indicates that the namespace supports Protection Information Type 1. Bit 0 if cleared to '0' indicates that the namespace does not support Protection Information Type 1.

Set Feature



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

DWord

0 Command Identifier, FUSE, Opcode

1 Namespace Identifier

2

3

4

5

6–7 PRP Entry 1

8–9 PRP Entry 2

10 Feature Identifier

11 Parameter

12

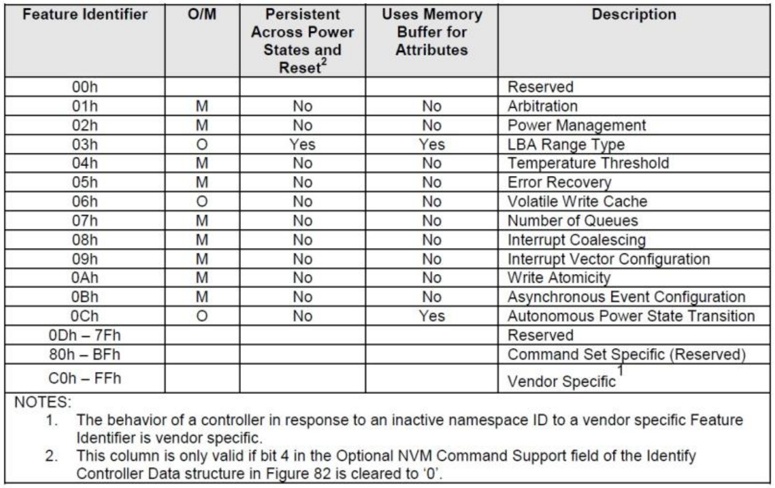
13

14

15

Set value of configurable feature

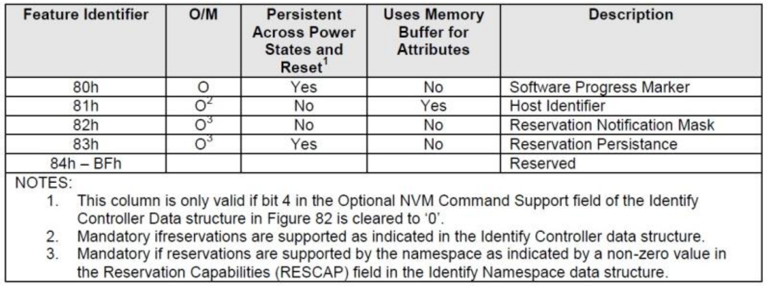
- **PRP Entry 1** – Starting address of where Feature data is located (used by some features)
- **PRP Entry 2** – Starting address of where remainder of where feature data is located (used by some features)
- **Parameter – Feature parameter** (used by some features**)**
- **Feature Identifier – ID of feature**



Feature Identifier	O/M	Persistent Across Power States and Reset[2]	Uses Memory Buffer for Attributes	Description
00h				Reserved
01h	M	No	No	Arbitration
02h	M	No	No	Power Management
03h	O	Yes	Yes	LBA Range Type
04h	M	No	No	Temperature Threshold
05h	M	No	No	Error Recovery
06h	O	No	No	Volatile Write Cache
07h	M	No	No	Number of Queues
08h	M	No	No	Interrupt Coalescing
09h	M	No	No	Interrupt Vector Configuration
0Ah	M	No	No	Write Atomicity
0Bh	M	No	No	Asynchronous Event Configuration
0Ch	O	No	Yes	Autonomous Power State Transition
0Dh – 7Fh				Reserved
80h – BFh				Command Set Specific (Reserved)
C0h – FFh				Vendor Specific[1]

NOTES:

1. The behavior of a controller in response to an inactive namespace ID to a vendor specific Feature Identifier is vendor specific.
2. This column is only valid if bit 4 in the Optional NVM Command Support field of the Identify Controller Data structure in Figure 82 is cleared to ‘0’.



Feature Identifier	O/M	Persistent Across Power States and Reset[1]	Uses Memory Buffer for Attributes	Description
80h	O	Yes	No	Software Progress Marker
81h	O[2]	No	Yes	Host Identifier
82h	O[3]	No	No	Reservation Notification Mask
83h	O[3]	Yes	No	Reservation Persistance
84h – BFh				Reserved

NOTES:

1. This column is only valid if bit 4 in the Optional NVM Command Support field of the Identify Controller Data structure in Figure 82 is cleared to ‘0’.
2. Mandatory ifreservations are supported as indicated in the Identify Controller data structure.
3. Mandatory if reservations are supported by the namespace as indicated by a non-zero value in the Reservation Capabilities (RESCAP) field in the Identify Namespace data structure.

Get Log Page

DWord	Byte 3	Byte 2	Byte 1	Byte 0
0	Command Identifier		FUSE	Opcode
1	Namespace Identifier			
2				
3				
4				
5				
6	PRP Entry 1			
7				
8	PRP Entry 2			
9				
10		Number of DWords		Log Page Identifier
11				
12				
13				
14				
15				

Retrieves up to 4KB of data from specified "log page"

- **PRP Entry 1** – Starting address of where log page should be written
- **PRP Entry 2** – Starting address of where remainder of remainder of log page should be written
- **Number of DWords –** Number of DWords to transfer
- **Log Page Identifier –** ID of log page to retrieve

Log Identifier	O/M	Description
00h		Reserved
01h	M	Error Information
02h	M	SMART / Health Information
03h	M	Firmware Slot Information
04h – 7Fh		Reserved
80h – BFh		I/O Command Set Specific
C0h – FFh		Vendor specific

O/M: O = Optional, M = Mandatory

Asynchronous Event Request

	Byte 3								Byte 2								Byte 1								Byte 0							
DWord	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																																
5																																
6																																
7																																
8																																
9																																
10																																
11																																
12																																
13																																
14																																
15																																

	Byte 3								Byte 2								Byte 1								Byte 0							
DWord	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									Log Page								Aync Event Info													Type		
1																																
2	SQ Identifier																SQ Head Pointer															
3	Status Field															P	Command Identifier															

Method to obtain asynchronous event status from controller

Event signaled by a completion to a previously issued asynchronous event request command

After asynchronous event, events of that same type are masked until the host reads the corresponding log page

- **Type** – Type of asynchronous event
  - Error Status
  - SMART / Health status
  - Vendor Specific
- **Async Event Info** – Provides error type specific details
  - Examples:
    - Temperature above threshold
    - Spare space below threshold
    - Invalid doorbell value write
- **Log Page –** ID of log page to retrieve more information and clear mask (using Get Log Page)

Controller Initialization (Part 2)

The host performs the following actions in sequence to initialize the controller to begin executing IO commands:

1. determine the controller configuration using the Identify command (Controller data structure)
2. determine namespace configuration for each namespace by using the Identify command (Namespace data structure)
3. determine the number of I/O Submission and Completion Queues supported using the Set Features command.
4. After determining the number of I/O Queues, the MSI and/or MSI-X registers should be configured.
5. allocate the appropriate number of I/O Completion Queues, using the Create I/O Completion Queue command
6. allocate the appropriate number of I/O Submission Queues, using the Create I/O Submission Queue command
7. If the host desires asynchronous notification of error or health events, submit an appropriate number of Asynchronous Event Request commands.

NVM Cmd Set Admin Commands

Command	Required or Optional	Category
Format NVM	Optional	Admin
Security Send	Optional	
Security Receive	Optional	

NVM Command Set

Command	Required or Optional	Category
Read	Required	Required Data Commands
Write	Required	
Flush	Required	
Write Uncorrectable	Optional	Optional Data Commands
Write Zeros	Optional	
Compare	Optional	
Dataset Management	Optional	Data Hints
Reservation Acquire	Optional	Reservations Commands
Reservation Register	Optional	
Reservation Release	Optional	
Reservation Report	Optional	
Vendor Specific Commands	Optional	Vendor Specific

NVM commands support both PRPs or SGLs.

Read

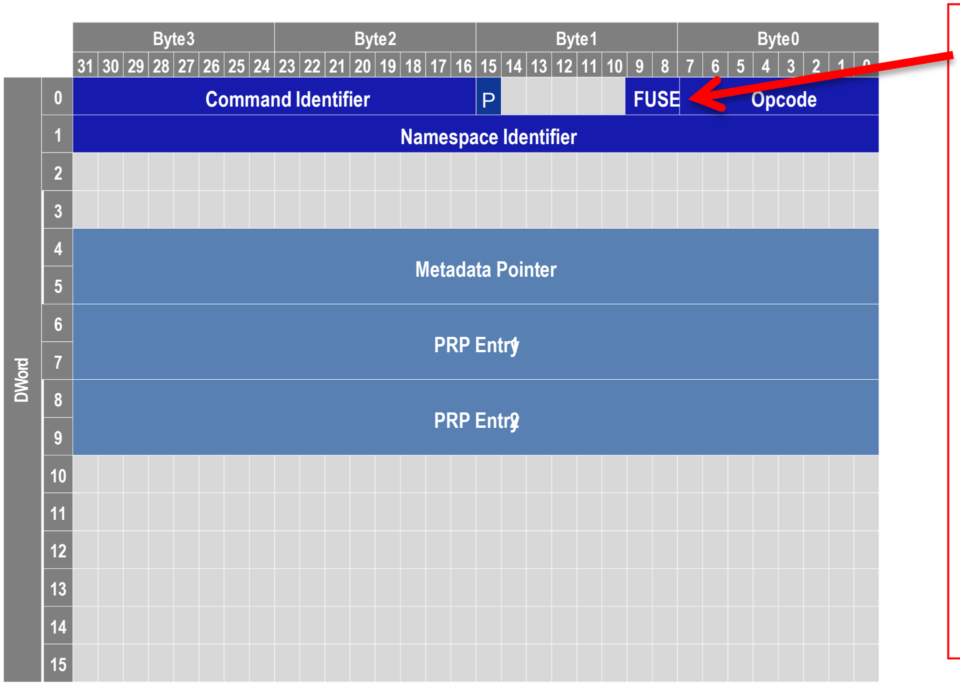
DWord	Byte 3	Byte 2	Byte 1	Byte 0
	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0	Command Identifier		P FUSE	Opcode
1	Namespace Identifier			
2				
3				
4	Metadata Pointer or Metadata SGL Segment Pointer			
5				
6	PRP Entry1			
7				
8	PRP Entry2			
9				
10	Starting LBA			
11				
12	LR FUA PRINFO		Number of Logical Blocks	
13				DSM
14	Expected Initial Logical Block Reference Tag			
15	Expected Logical Block Application Tag		Expected Logical Block Application Tag Mask	

Read logical blocks from NVM and perform specified protection information processing

- **PRP Entry 1, PRP Entry 2, Metadata Pointer** – Host buffers to write data read from NVM
- **Starting LBA** – Address of first logical block to read
- **Number of Logical Blocks** – Number of logical blocks to read from NVM
- **Protection Information Field (PRINFO)**
  - **Protection Information Action**
    - **Pass protection information or read and strip**
  - **Protection Information Check**
    - **Guard field check or no check**
    - **Application tag field check or no check**
    - **Reference tag field check or no check**
- **Force Unit Access (FUA)**
  - Return data from NVM
- **Limited Retry (LR)**
  - Apply limited retry or apply all available error recovery means to return data
- **Data Set Management (DSM)**
  - **Described later**
- **Protection Information Related Fields**
  - **Expected Initial Logical Block Reference Tag**
  - **Expected Logical Block Application Tag Mask**
  - **Expected Logical Block Application Tag**

Fused Operation

- A fused operation is a method to create a complex command by "fusing" together two simpler commands.



DWord	Byte3	Byte2	Byte1			Byte0
	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		P		FUSE	Opcode
1	Namespace Identifier					
2						
3						
4	Metadata Pointer					
5						
6	PRP Entry1					
7						
8	PRP Entry2					
9						
10						
11						
12						
13						
14						
15						

- This field specifies whether this command is part of a fused operation and if so, which command it is in the sequence.

- Field definition

  00b Normal operation

  01b Fused operation, first command

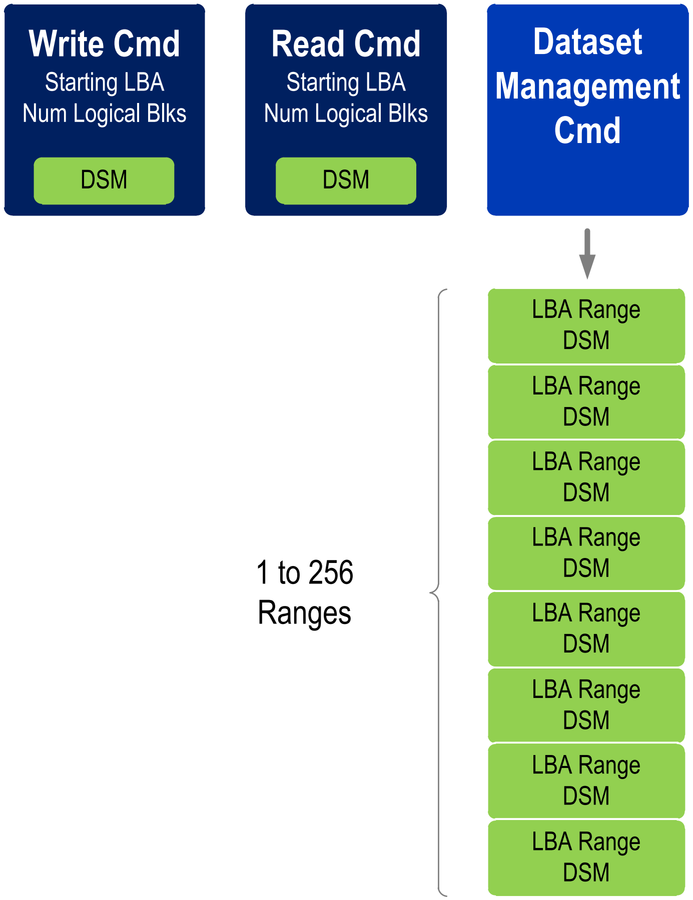
  10b Fused operation, second command

  11b Reserved

Compare and Write

- Compare and write is the only defined fused operation
- Compare and Write commands are submitted in adjacent slots in the submission queue
- Compare and Write are executed as atomic unit
  - A completion queue entry is posted for each of the two commands
  - If Compare succeeds, then Write command is executed
  - If Compare fails, then Write command is aborted
    - “Command Aborted due to Failed Fused Command” completion status for write command
  - Both Compare and Write must operate on the same LBA range

Data Set Management (DSM) Hints

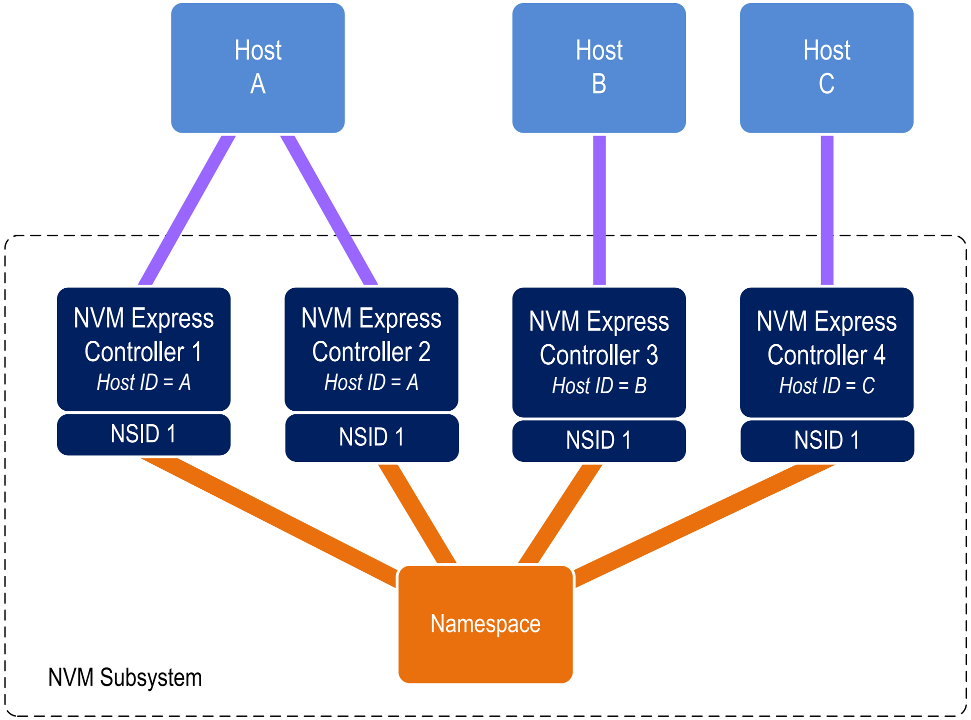


- DSM Hints
  - Access size (in logical blocks)
  - Written in near future
  - Sequential read
  - Sequential write
  - Access latency (longer, typical, small)
  - Access frequency
    - Typical read and write
    - Infrequent read and write
    - Infrequent write, frequent read
    - Frequent write, infrequent read
    - Frequent read and write
- Dataset Management Command
  - Deallocate (“TRIM”)
  - Integral write dataset
  - Integral read dataset

Reservation Overview

- Reservations provide capabilities that may be utilized by two or more hosts to provide coordinated access to a shared namespace
  - The protocol and manner in which these capabilities are used are outside the scope of NVMe
  - Reservations are functionally compatible with T10 persistent reservations
- Reservations are on a namespace and restrict host access to that namespace
  - If a host submits a command to a namespace in the presence of a reservation and lacks sufficient rights, then the command is aborted by the controller with a status of Reservation Conflict
- Capabilities are provided to allow recovery from a reservation held by a failing or uncooperative host

Example Multi-Host System



Host Identifier (Host ID) associated with each controller allows NVM subsystem to identify controllers associated with the same host and preserve reservation properties across controllers

New NVM Reservation Commands

NVM I/O Command	Operation
Reservation Register	• Register a reservation key • Unregister a reservation key • Replace a reservation key
Reservation Acquire	• Acquire a reservation on a namespace • Preempt a reservation held on a namespace • Abort a reservation held on a namespace
Reservation Release	• Release a reservation held on a namespace • Clear a reservation held on a namespace
Reservation Report	• Retrieve reservation status data structure Type of reservation held on the namespace (if any) Persist through power loss state Reservation status, Host ID, reservation key for each host that has access to the namespace

Command Behavior In Presence of a Reservation

Reservation Type	Reservation Holder		Registrant		Non-Registrant		Reservation Holder Definition
	Read	Write	Read	Write	Read	Write	
Write Exclusive	Y	Y	Y	N	Y	N	One Reservation Holder
Exclusive Access	Y	Y	N	N	N	N	One Reservation Holder
Write Exclusive - Registrants Only	Y	Y	Y	Y	Y	N	One Reservation Holder
Exclusive Access - Registrants Only	Y	Y	Y	Y	N	N	One Reservation Holder
Write Exclusive - All Registrants	Y	Y	Y	Y	Y	N	All Registrants are Reservation Holders
Exclusive Access - All Registrants	Y	Y	Y	Y	N	N	All Registrants are Reservation Holders

Reservation Acquire

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		P FUSE	Opcode
1	Namespace Identifier			
2				
3				
4				
5				
6	PRP Entry 1			
7				
8	PRP Entry 2			
9				
10			Reservation Type	IEKEY RACQA
11				
12				
13				
14				
15				

The Reservation Acquire command is used to acquire a reservation on a namespace, preempt a reservation held on a namespace, and abort a reservation held on a namespace

- **Reservation Type (RTYPE) -** specifies the type of reservation to be created
- **Ignore Existing Key (IEKEY):** If this bit is set to a ‘1’, then the Current Reservation Key (CRKEY) check is disabled and the command shall succeed regardless of the CRKEY field value
- **Reservation Acquire Action (RACQA):** specifies the action that is performed by the command.

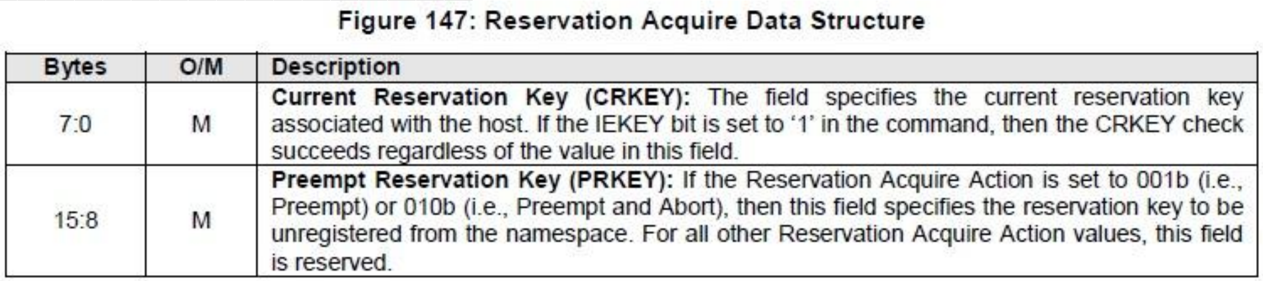
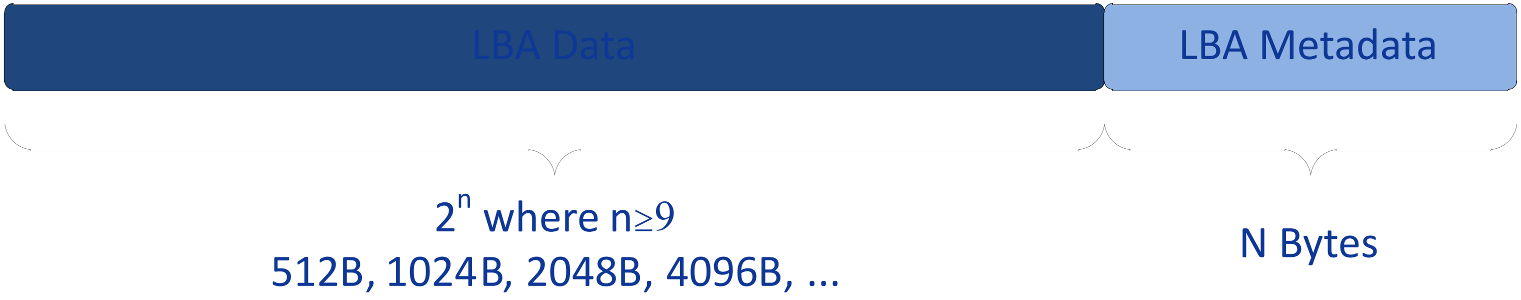


Figure 147: Reservation Acquire Data Structure

Bytes	O/M	Description
7:0	M	**Current Reservation Key (CRKEY):** The field specifies the current reservation key associated with the host. If the IEKEY bit is set to ‘1’ in the command, then the CRKEY check succeeds regardless of the value in this field.
15:8	M	**Preempt Reservation Key (PRKEY):** If the Reservation Acquire Action is set to 001b (i.e., Preempt) or 010b (i.e., Preempt and Abort), then this field specifies the reservation key to be unregistered from the namespace. For all other Reservation Acquire Action values, this field is reserved.

Logical Block Format



LBA Data

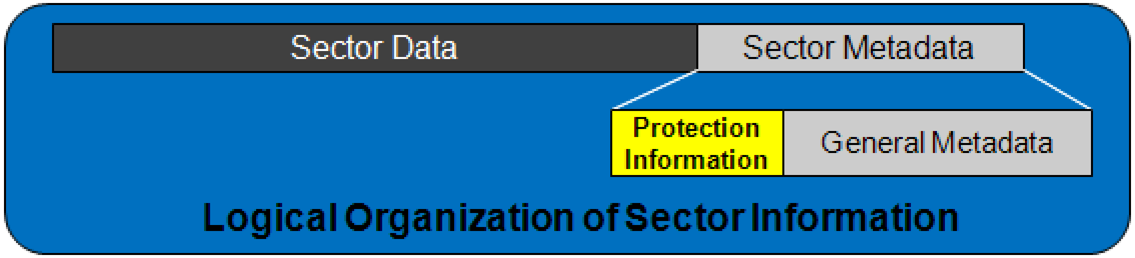
LBA Metadata

2^n where $n \geq 9$
512B, 1024B, 2048B, 4096B, ...

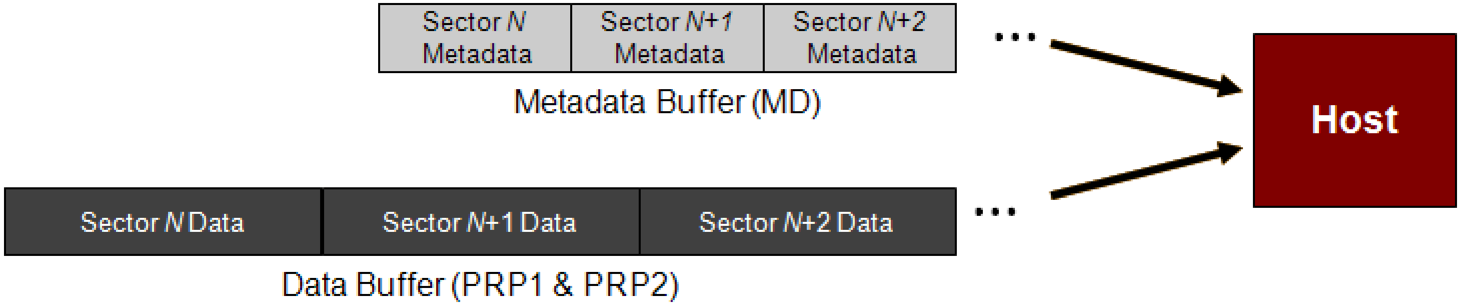
N Bytes

- Identify Namespace data structure indicates supported formats
  - A Namespace may indicate support for up to 16 different formats
    - Example:
      – 512b, 520b, 528b, 4096b, …

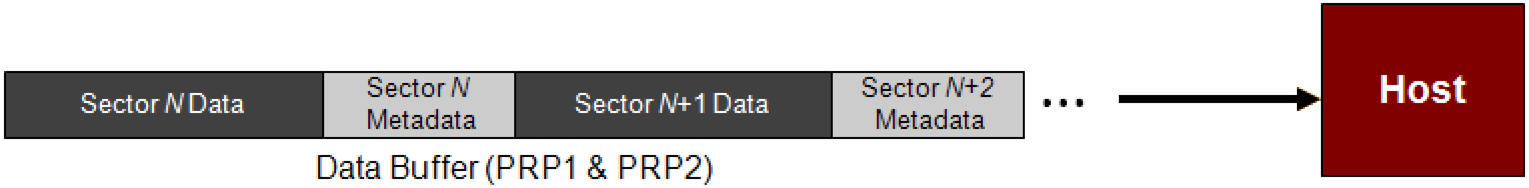
Metadata Host Transfer Options


Sector Data
Sector Metadata
Protection Information
General Metadata

Logical Organization of Sector Information

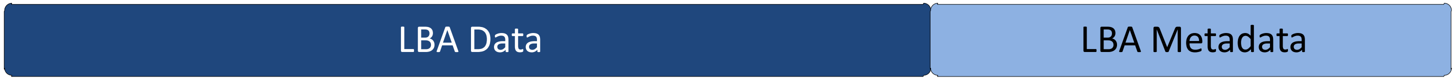

Sector *N* Metadata
Sector *N+1* Metadata
Sector *N+2* Metadata
…
Metadata Buffer (MD)
Host
Sector *N* Data
Sector *N*+1 Data
Sector *N*+2 Data
…
Data Buffer (PRP1 & PRP2)

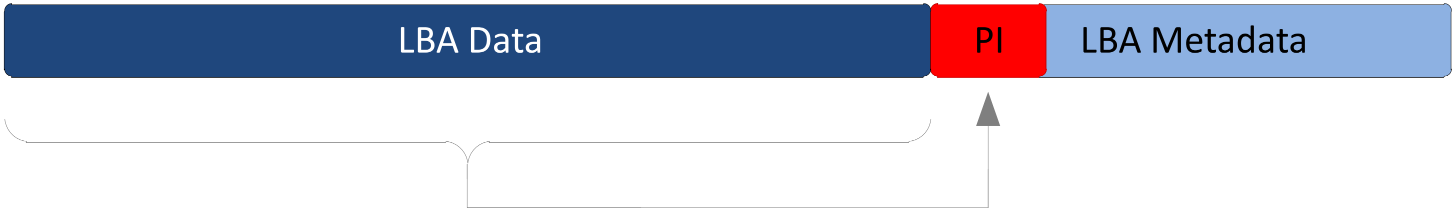
Data with Metadata in Separate Contiguous Buffer, "DIX Like"


Sector *N* Data
Sector *N* Metadata
Sector *N*+1 Data
Sector *N*+2 Metadata
…
Host
Data Buffer (PRP1 & PRP2)

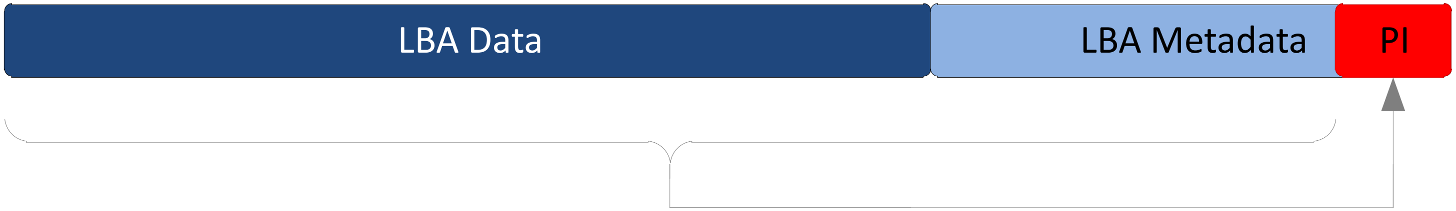
Metadata as Part of Sector Data, "DIF like"

Protection Information Location



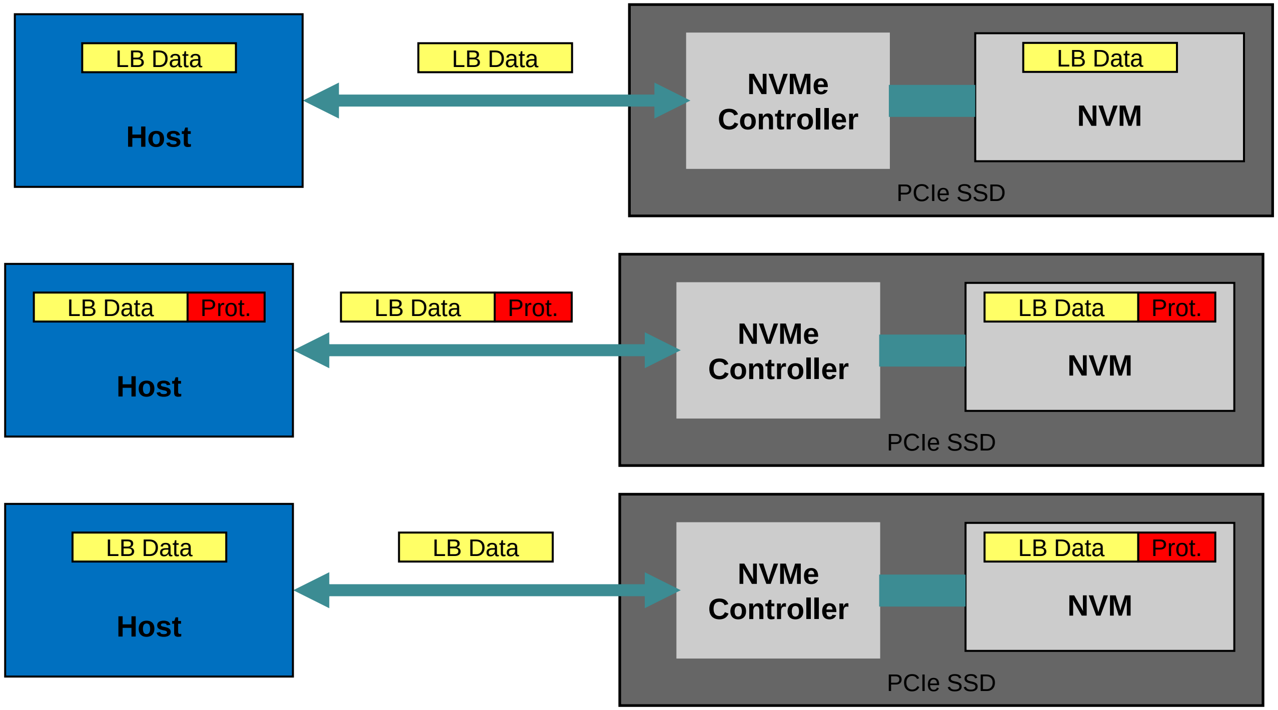


Protection Information in First 8B of Metadata



Protection Information in Last 8B of Metadata

End-to-End Data Protection Options



No Data Protection Information

End-to-End Data Protection Information

"Insert" & "Strip" End-to-End Data Protection Information

- Functionally compatible with T10 DIF & DIX, including DIF Type 1, 2, and 3
- End-to-end protection configured per namespace with NVM Format command
- Controller may "insert" and "strip" protection information

Format NVM

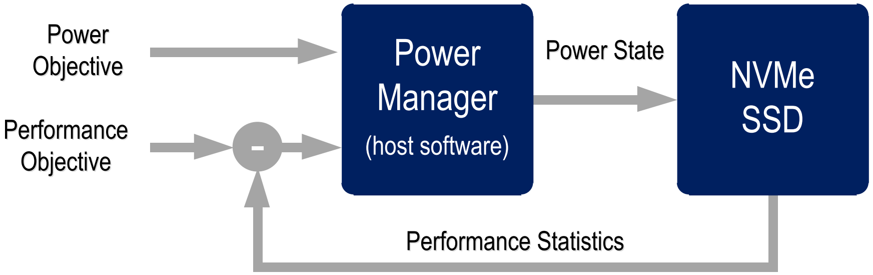
		Byte 3								Byte 2								Byte 1								Byte 0							
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DWord	0	Command Identifier																						FUSE		Opcode							
	1	Namespace Identifier																															
	2																																
	3																																
	4																																
	5																																
	6																																
	7																																
	8																																
	9																																
	10																					SES			PIL	PI			MS	LBAF			
	11																																
	12																																
	13																																
	14																																
	15																																

Used to low level format a namespace

- **Support for this command is optional**
- **May apply to a specific namespace or to all namespaces**

- **LBA Format (LBAF)** – Indicates one of the supported LBA formats (in Identify)
- **Metadata Settings (MS)** – Extended LBA or two buffers
  - 0 – Two buffers
  - 1 – Extended LBA
- **Protection Information (PI)** – Protection information mode
  - 0 – No PI
  - 1 – Type 1
  - 2 – Type 2
  - 3 – Type 3
- **Protection Information Location (PIL)**
  - 0 – Last 8 bytes of metadata
  - 1 – First 8 bytes of metadata
- **Secure Erase Settings (SES)**
  - 0 – No secure erase
  - 1 – User data erase
  - 2 – Cryptographic erase

NVMe Power Management



Power State Descriptor Table

Power State	Maximum Power	Operational State	Entry Latency	Exit Latency	Relative Read Throughput	Relative Read Latency	Relative Write Throughput	Relative Write Latency
0	25 W	Yes	5 µs	5 µs	0	0	0	0
1	18 W	Yes	5 µs	7 µs	0	0	1	0
2	18 W	Yes	5 µs	8 µs	1	0	0	0
3	15 W	Yes	20 µs	15 µs	2	1	2	1
4	7 W	Yes	20 µs	30 µs	1	2	3	1
5	1 W	No	100 mS	50 mS	-	-	-	-
6	.25 W	No	100 mS	500 mS	-	-	-	-

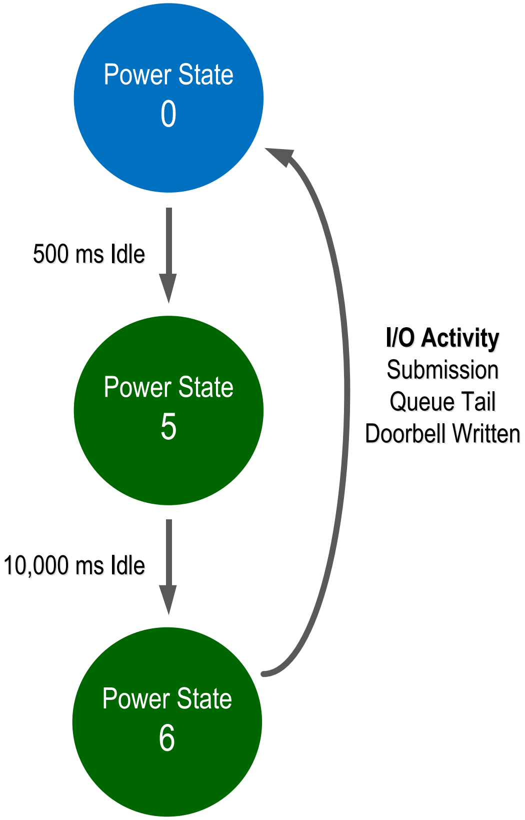
Autonomous Power State Transitions

Power State Descriptor Table

Power State	Maximum Power	Operational State	Entry Latency	Exit Latency
0	25 W	Yes	5 µs	5 µs
1	18 W	Yes	5 µs	7 µs
2	18 W	Yes	5 µs	8 µs
3	15 W	Yes	20 µs	15 µs
4	7 W	Yes	20 µs	30 µs
5	1 W	No	100 mS	50 mS
6	.25 W	No	100 mS	500 mS

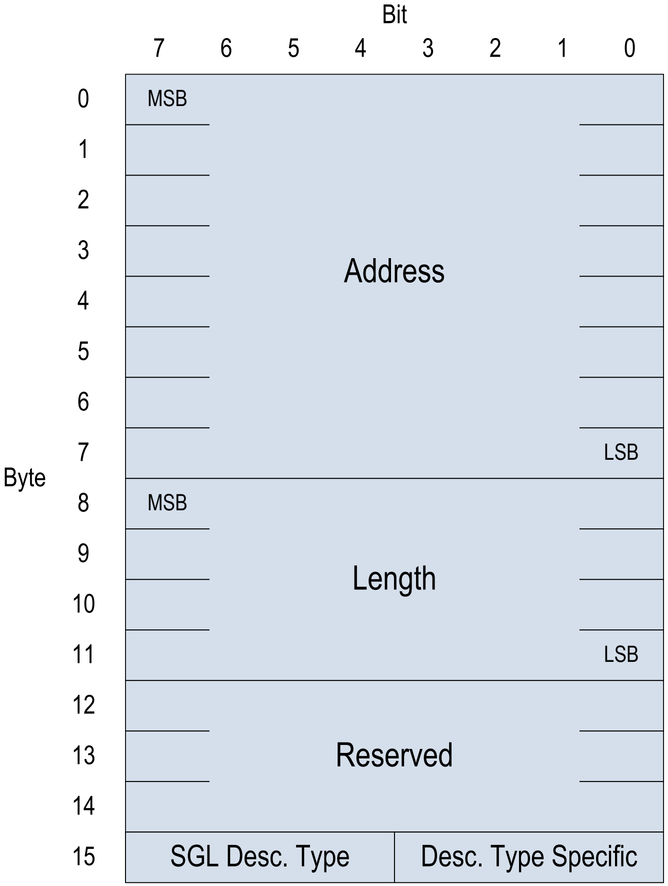
Autonomous Power State Transition Table

Idle Time Prior to Transition	Idle Transition Power State
500 ms	5
500 ms	5
500 ms	5
500 ms	5
500 ms	5
10,000 ms	6
-	-



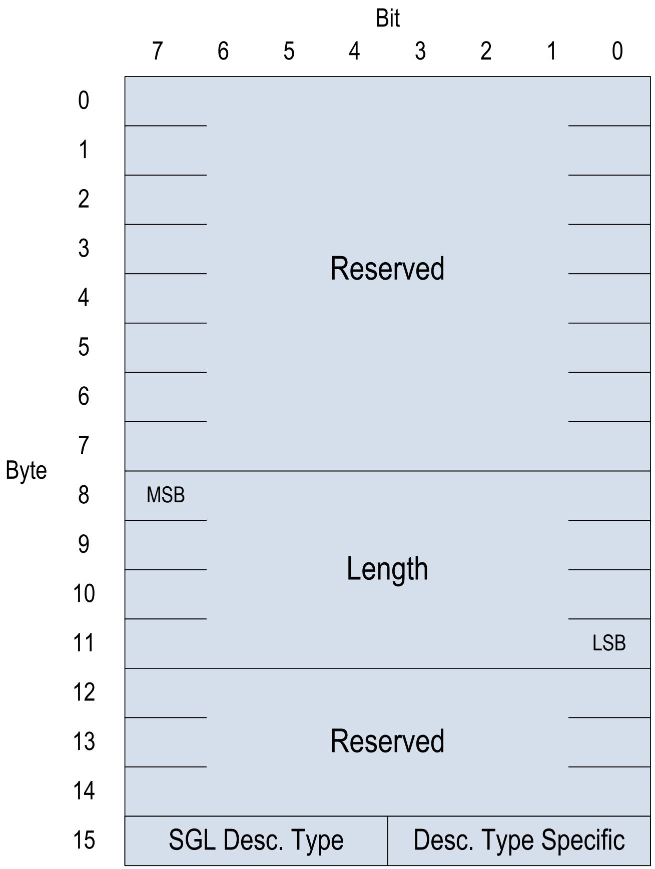
Backup

SGL Data Block Descriptor



- Used to transfer data between PCIe memory and Controller
- Address
  - 64-bit PCIe address of the data
  - Supports any byte alignment
- Length
  - Length of the data block in bytes
  - A value of zero indicates that no data is transferred

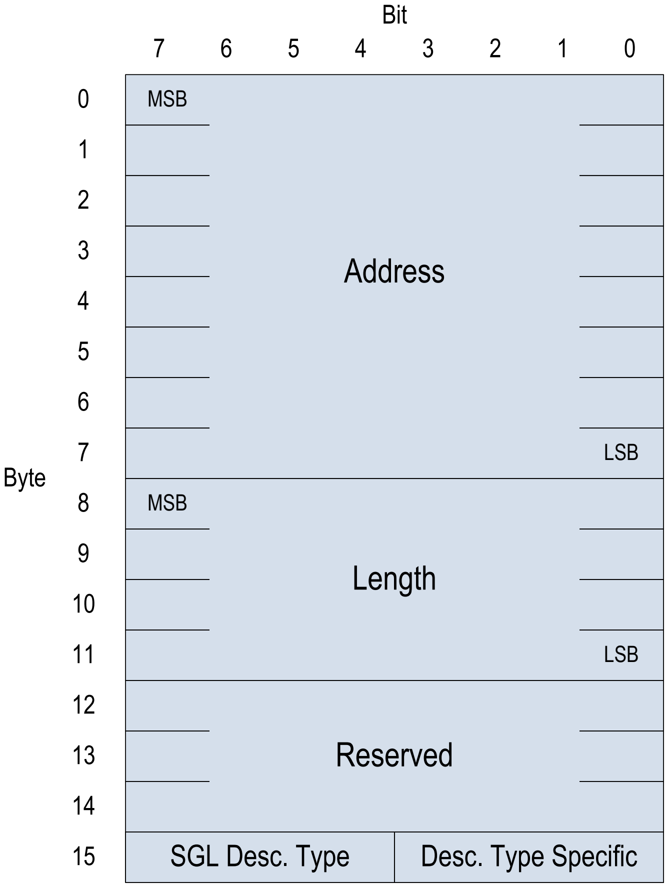
SGL Bit Bucket Descriptor



	Bit							
Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2								
3								
4								
5								
6								
7								
8	Length (MSB)							
9								
10								
11	(LSB)							
12	Reserved							
13								
14								
15	SGL Desc. Type				Desc. Type Specific			

- Skip source data bytes
  - Only makes sense for controller to host transfers
  - This descriptor is ignored in host to controller transfers
- Length
  - Length of the data block in bytes
  - A value of zero indicates that no data is transferred

SGL Segment and SGL Last Segment Descriptors



	Bit							
Byte	7	6	5	4	3	2	1	0
0	MSB			Address				
1								
2								
3								
4								
5								
6								
7								LSB
8	MSB			Length				
9								
10								
11								LSB
12				Reserved				
13								
14								
15	SGL Desc. Type				Desc. Type Specific			

- SGL Segment - Pointer to next SGL Segment
- SGL Last Segment - Pointer to last SGL Segment
- Address
  - Address in PCIe memory of next segment
  - Must be 64-bit aligned
- Length
  - Length of the segment in bytes
  - Must be multiple of 16 (a descriptor is 16B)

Delete I/O Submission Queue

		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
DWord	0	Command Identifier																						FUSE		Opcode							
	1	Namespace Identifier																															
	2																																
	3																																
	4																																
	5																																
	6																																
	7																																
	8																																
	9																																
	10																	Queue Identifier															
	11																																
	12																																
	13																																
	14																																
	15																																

Delete specified I/O submission queue

- **Queue Identifier** – Submission queue ID number
- **Command Specific Error Values**
  - Invalid Queue Identifier

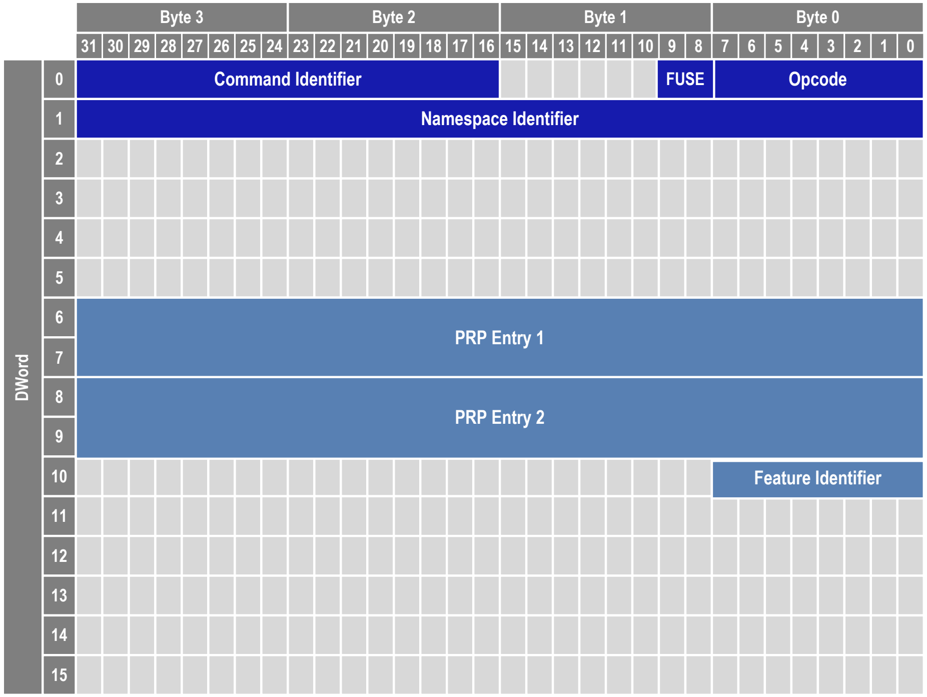
Delete I/O Completion Queue

		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
DWord	0	Command Identifier																						FUSE		Opcode							
	1	Namespace Identifier																															
	2																																
	3																																
	4																																
	5																																
	6																																
	7																																
	8																																
	9																																
	10																	Queue Identifier															
	11																																
	12																																
	13																																
	14																																
	15																																

Delete specified I/O completion queue

- **Queue Identifier** – Completion queue ID number

Get Feature



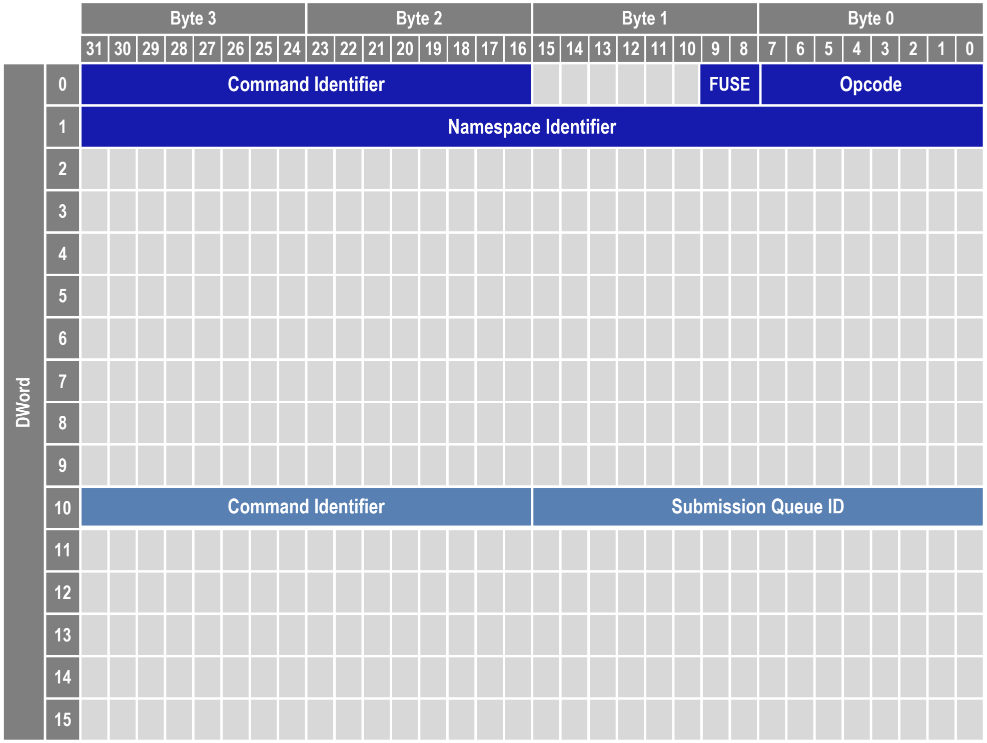
		Byte 3								Byte 2								Byte 1								Byte 0							
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DWord	0	Command Identifier																						FUSE		Opcode							
	1	Namespace Identifier																															
	2																																
	3																																
	4																																
	5																																
	6	PRP Entry 1																															
	7																																
	8	PRP Entry 2																															
	9																																
	10																									Feature Identifier							
	11																																
	12																																
	13																																
	14																																
	15																																

Get value of configurable feature

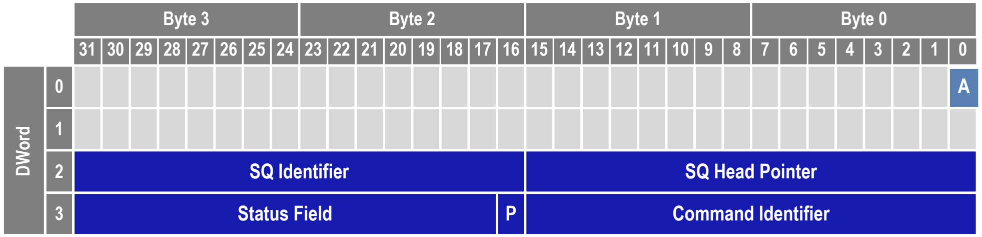
- **PRP Entry 1** – Starting address of where feature data should be written (used by some features)
- **PRP Entry 2** – Starting address of where remainder of where feature data should be written (used by some features)
- **Feature Identifier –** ID of feature

- Feature value returned in memory (PRPs) or in DWord 0 of completion entry

Abort



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																																
5																																
6																																
7																																
8																																
9																																
10	Command Identifier																Submission Queue ID															
11																																
12																																
13																																
14																																
15																																



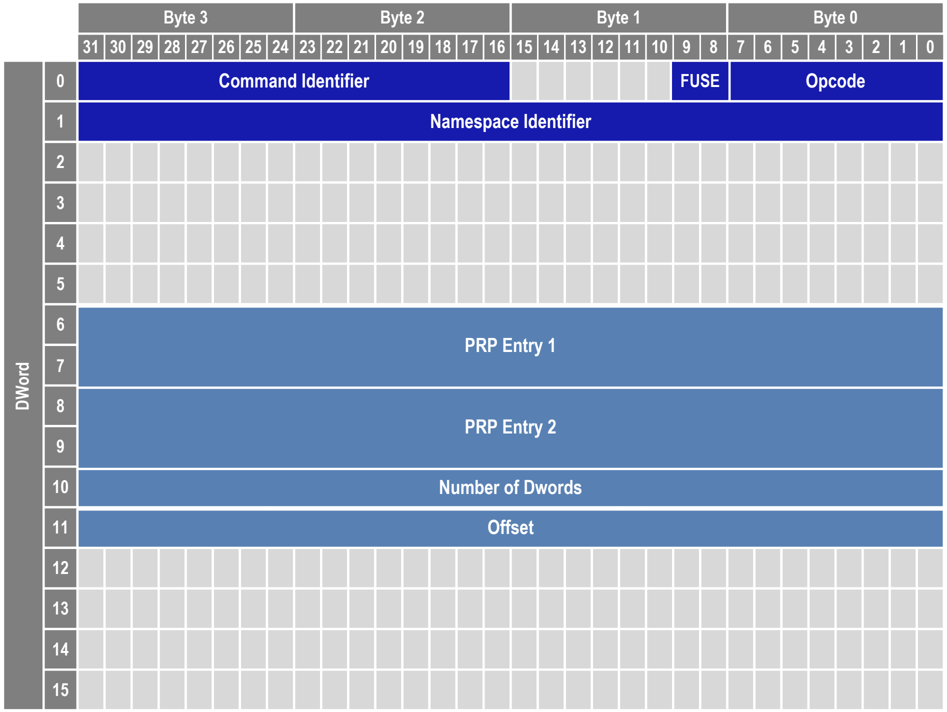
DWord	Byte 3								Byte 2								Byte 1								Byte 0							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																																A
1																																
2	SQ Identifier																SQ Head Pointer															
3	Status Field															P	Command Identifier															

Used to cancel/abort a specific command previously issued on the admin or I/O submission queue

- **(Submission Queue ID, Command Identifier) is globally unique**
- **The aborting of a command is best effort by the controller**
- **Implementation specific when a controller completes the command when the command is not found**
- **A controller specifies the maximum number of outstanding abort command that it can support in Identify Controller Data Structure**

- **Submission Queue ID**– ID of submission queue on which command was issued
- **Command Identifier**– ID of the command to abort
- **A**– Abort status
  - 0 – Command was aborted
  - 1 – Command was not aborted

Firmware Image Download



	Byte 3								Byte 2								Byte 1								Byte 0							
DWord	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																																
5																																
6	PRP Entry 1																															
7																																
8	PRP Entry 2																															
9																																
10	Number of Dwords																															
11	Offset																															
12																																
13																																
14																																
15																																

Used to download all or portion of a firmware image

- **Firmware image may consist of multiple pieces**
- **Pieces do not need to be do downloaded in order**
- **Pieces must not overlap**

- **PRP Entry 1 and PRP Entry 2** – PRP entries / list pointer where firmware piece is located
- **Command Identifier**– ID of the command to abort
- **Number of Dwords**– Number of DWords contains in the portion of the firmware image being downloaded
- **Offset**– DWord offset from 0 (the start) associated with this firmware piece

Firmware Activate

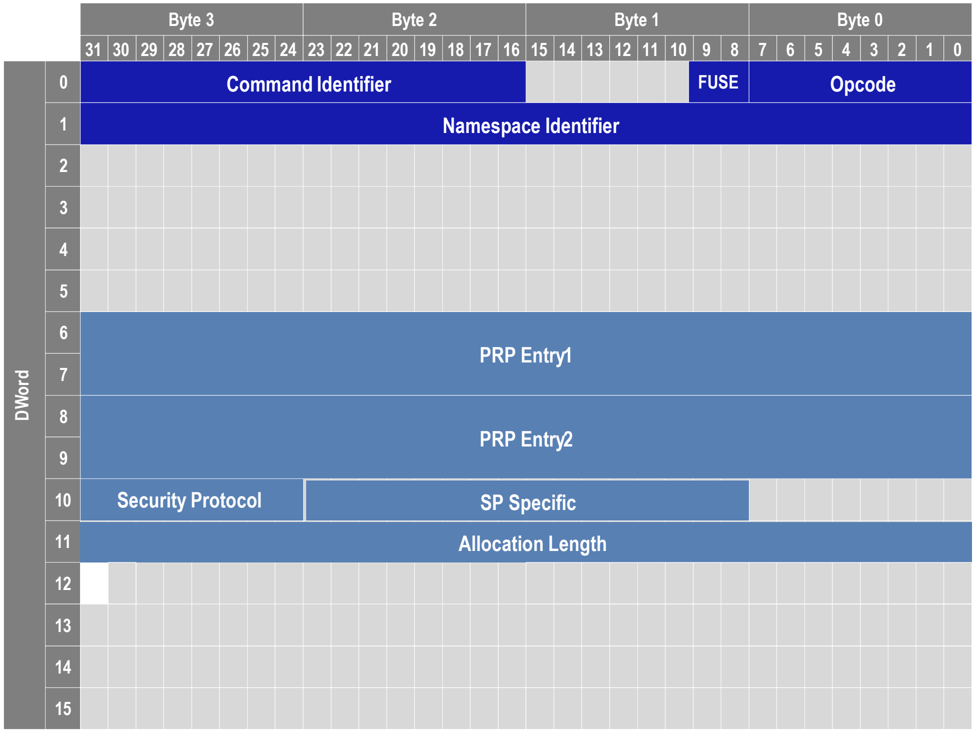
DWord	Byte 3								Byte 2								Byte 1								Byte 0							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Command Identifier																						FUSE		Opcode							
1	Namespace Identifier																															
2																																
3																																
4																																
5																																
6																																
7																																
8																																
9																																
10																												AA		FS		
11																																
12																																
13																																
14																																
15																																

Used to activate a firmware images

- **Newly activated image is the one that runs after a controller reset**
- **Performs two orthogonal operations**
  - **Validates and loads downloaded firmware image into firmware slot**
  - **Activates a firmware slot**

- **Active Action (AA)** – Action taken on the downloaded image or image associated with a firmware slot
  - 00b – Downloaded image becomes the new image in the firmware slot specified by the FS field. This image is NOT activated.
  - 01b – Downloaded image becomes the new image in the firmware slot specified by the FS field and this image is activated.
  - 11b – Image contained in the firmware slot specified by the FS field is activated
- **Firmware Slot (FS)** – Field used by AA field to indicate which slot to be updated and/or activated
  - Values 1 through 7 indicate a slot number
  - Value of 0 indicates that the controller should pick a slot number

Security Received

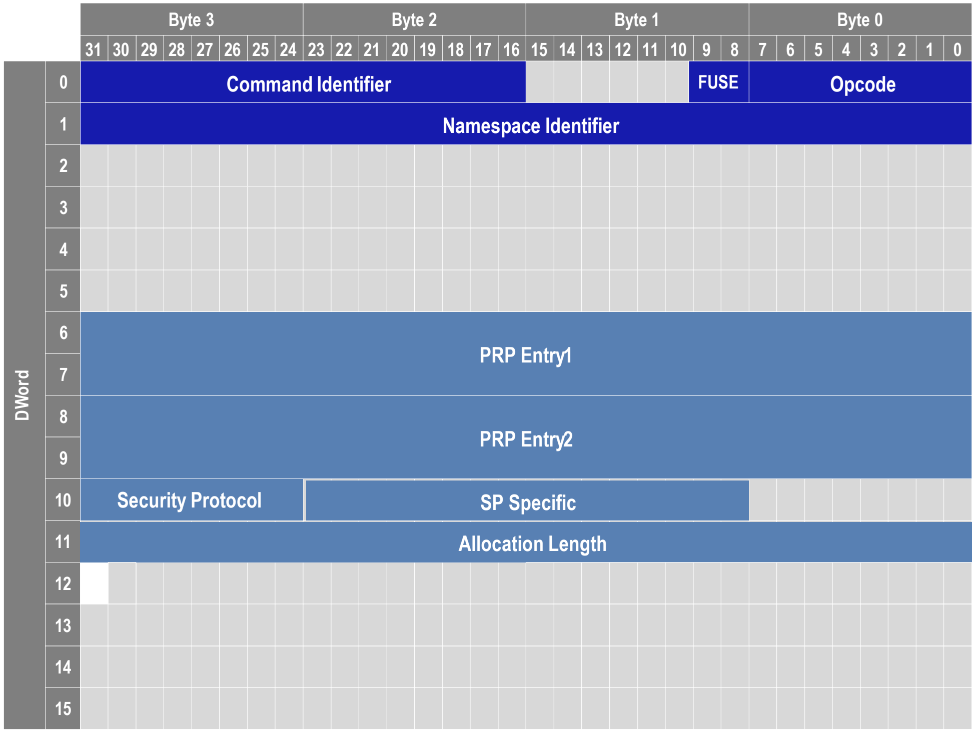


		Byte 3								Byte 2								Byte 1								Byte 0							
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DWord	0	Command Identifier																						FUSE		Opcode							
	1	Namespace Identifier																															
	2																																
	3																																
	4																																
	5																																
	6	PRP Entry1																															
	7																																
	8	PRP Entry2																															
	9																																
	10	Security Protocol								SP Specific																							
	11	Allocation Length																															
	12																																
	13																																
	14																																
	15																																

transfers the status and data result of one or more Security Send commands that were previously submitted to the controller

- **PRP Entry 1, PRP Entry 2 -** Host buffers that contains the security protocol information
- **Starting LBA** – Address of first logical block to read
- **Security Protocol**– specifies the security protocol as defined in SPC-4
- **SP Specific** - specific to the Security Protocol as defined in SPC-4
- **Allocation Length** - specific to the Security Protocol as defined in SPC-4**.**

Security Send



		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
DWord	0	Command Identifier																						FUSE		Opcode							
	1	Namespace Identifier																															
	2																																
	3																																
	4																																
	5																																
	6	PRP Entry1																															
	7																																
	8	PRP Entry2																															
	9																																
	10	Security Protocol								SP Specific																							
	11	Allocation Length																															
	12																																
	13																																
	14																																
	15																																

used to transfer security protocol data to the controller.

PRP Entry 1, PRP Entry 2 - Host buffers that contains the security protocol information

- **Starting LBA** – Address of first logical block to read
- **Security Protocol**– specifies the security protocol as defined in SPC-4
- **SP Specific** - specific to the Security Protocol as defined in SPC-4
- **Transfer Length -** specific to the Security Protocol as defined in SPC-4

Write

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		P FUSE	Opcode
1	Namespace Identifier			
2				
3				
4	Metadata Pointer or Metadata SGL Segment Pointer			
5				
6	PRP Entry1			
7				
8	PRP Entry2			
9				
10	Starting LBA			
11				
12	LR FUA PRINFO		Number of Logical Blocks	
13				DSM
14	Initial Logical Block Reference Tag			
15	Logical Block Application Tag		Logical Block Application Tag Mask	

Write logical blocks to NVM and perform specified protection information processing

- **PRP Entry 1, PRP Entry 2, Metadata Pointer** – Host buffers for read data to be written to NVM
- **Starting LBA** – Address of first logical block to written
- **Number of Logical Blocks** – Number of logical blocks to write to NVM
- **Protection Information Field (PRINFO)**
  - **Protection Information Action**
    - **Pass protection information or write and insert**
  - **Protection Information Check**
    - **Guard field check or no check**
    - **Application tag field check or no check**
    - **Reference tag field check or no check**
- **Force Unit Access (FUA)**
  - Write data to NVM
- **Limited Retry (LR)**
  - Apply limited retry or apply all available means to write data to NVM
- **Data Set Management (DSM)**
  - **Described later**
- **Protection Information Related Fields**
  - **Initial Logical Block Reference Tag**
  - **Logical Block Application Tag Mask**
  - **Logical Block Application Tag**

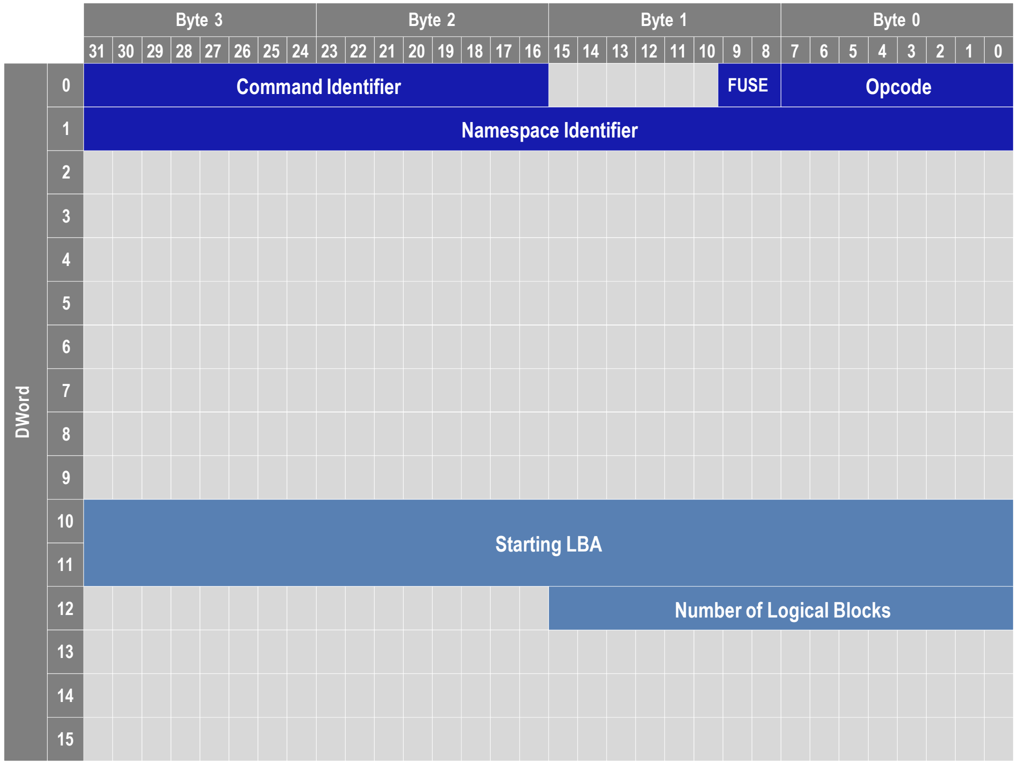
Flush

		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
DWord	0	Command Identifier																						FUSE		Opcode							
	1	Namespace Identifier																															
	2																																
	3																																
	4																																
	5																																
	6																																
	7																																
	8																																
	9																																
	10																																
	11																																
	12																																
	13																																
	14																																
	15																																

Causes any data in volatile storage to be flushed to non-volatile memory

- Volatile Write Cache (VWC) field in Indentify Controller Data Structure
  - 1 – Volatile write cache is present
    - Flush command may be used to write volatile data to NVM
    - Set Feature command may be used to enable/disable volatile write
  - 0 – Volatile write cache is NOT present

Write Uncorrectable



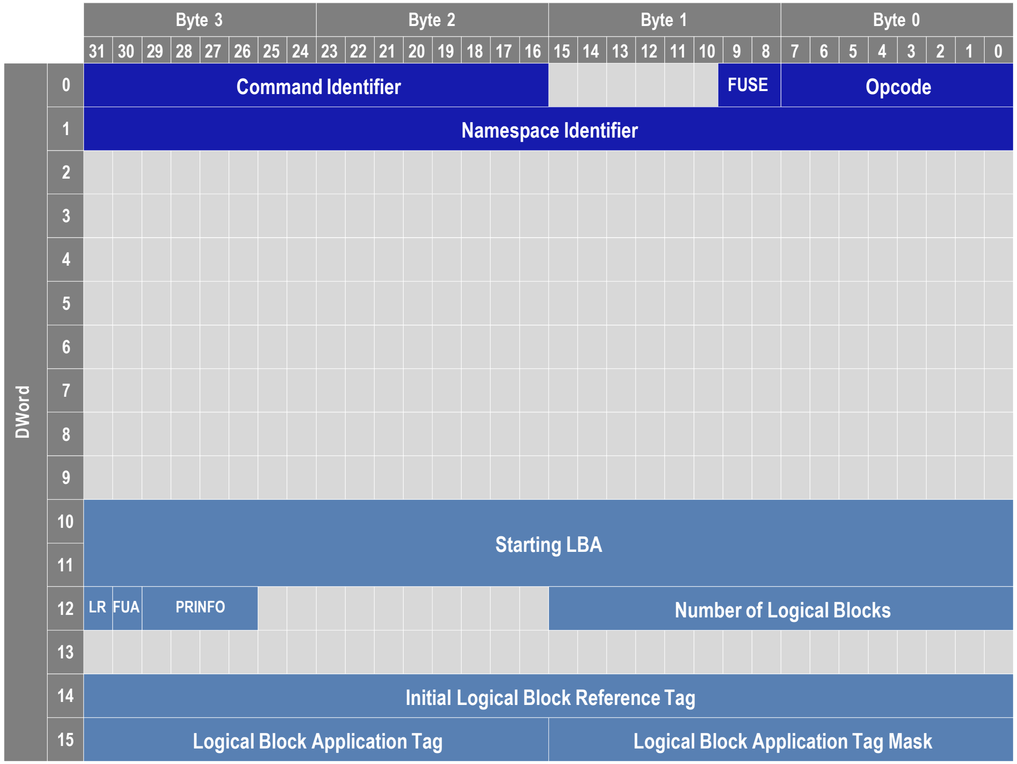
		Byte 3								Byte 2								Byte 1								Byte 0							
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DWord	0	Command Identifier																						FUSE		Opcode							
	1	Namespace Identifier																															
	2																																
	3																																
	4																																
	5																																
	6																																
	7																																
	8																																
	9																																
	10	Starting LBA																															
	11																																
	12																	Number of Logical Blocks															
	13																																
	14																																
	15																																

Mark logical blocks invalid

Subsequent read return “Unrecovered Read Error” status

- **Starting LBA** – Address of first logical block to written
- **Number of Logical Blocks** – Number of logical blocks to write to NVM

Write Zeroes

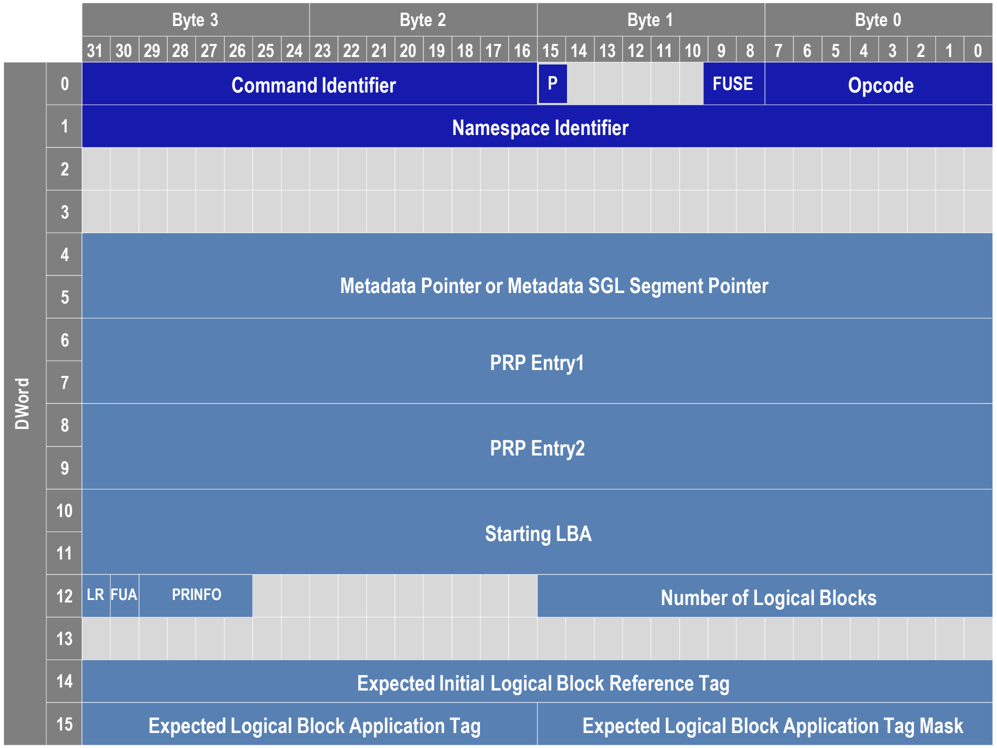


DWord	Byte 3								Byte 2								Byte 1								Byte 0							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Command Identifier																						FUSE		Opcode							
1	Namespace Identifier																															
2																																
3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

Write zeroes to the logical blocks on the NVM and perform specified protection information processing

- **Starting LBA** – Address of first logical block to written
- **Number of Logical Blocks** – Number of logical blocks to write to NVM
- **Protection Information Field (PRINFO)**
  - **Protection Information Action**
    - **Pass protection information or write and insert**
  - **Protection Information Check**
    - **Guard field check or no check**
    - **Application tag field check or no check**
    - **Reference tag field check or no check**
- **Force Unit Access (FUA)**
  - Write data to NVM
- **Limited Retry (LR)**
  - Apply limited retry or apply all available means to write data to NVM
- **Data Set Management (DSM)**
  - **Described later**
- **Protection Information Related Fields**
  - **Initial Logical Block Reference Tag**
  - **Logical Block Application Tag Mask**
  - **Logical Block Application Tag**

Compare

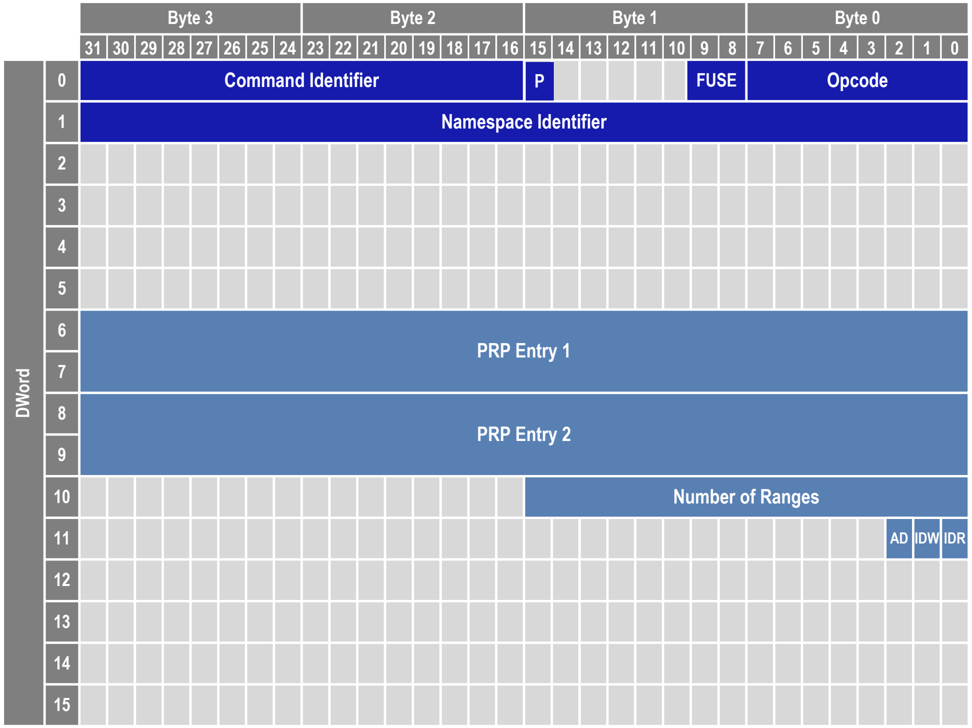


<table>
<tr><td></td><td></td><td colspan="8">Byte 3</td><td colspan="8">Byte 2</td><td colspan="8">Byte 1</td><td colspan="8">Byte 0</td></tr>
<tr><td></td><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td rowspan="16">DWord</td><td>0</td><td colspan="16">Command Identifier</td><td>P</td><td></td><td></td><td></td><td></td><td></td><td colspan="2">FUSE</td><td colspan="8">Opcode</td></tr>
<tr><td>1</td><td colspan="32">Namespace Identifier</td></tr>
<tr><td>2</td><td colspan="32"></td></tr>
<tr><td>3</td><td colspan="32"></td></tr>
<tr><td>4</td><td colspan="32" rowspan="2">Metadata Pointer or Metadata SGL Segment Pointer</td></tr>
<tr><td>5</td></tr>
<tr><td>6</td><td colspan="32" rowspan="2">PRP Entry1</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td><td colspan="32" rowspan="2">PRP Entry2</td></tr>
<tr><td>9</td></tr>
<tr><td>10</td><td colspan="32" rowspan="2">Starting LBA</td></tr>
<tr><td>11</td></tr>
<tr><td>12</td><td>LR</td><td>FUA</td><td colspan="4">PRINFO</td><td colspan="10"></td><td colspan="16">Number of Logical Blocks</td></tr>
<tr><td>13</td><td colspan="32"></td></tr>
<tr><td>14</td><td colspan="32">Expected Initial Logical Block Reference Tag</td></tr>
<tr><td>15</td><td colspan="16">Expected Logical Block Application Tag</td><td colspan="16">Expected Logical Block Application Tag Mask</td></tr>
</table>

Read logical block data from NVM and compare the data read to data buffer(s) fetched from the host

- Same fields as a read operation
  - No Dataset Management field
- Protection information checking is performed (if enabled)

Dataset Management



DWord	Byte 3 (31–24)	Byte 2 (23–16)	Byte 1 (15–8)	Byte 0 (7–0)
0	Command Identifier		P (15), FUSE (9–8)	Opcode
1	Namespace Identifier			
2				
3				
4				
5				
6	PRP Entry 1			
7				
8	PRP Entry 2			
9				
10			Number of Ranges	
11				AD (2), IDW (1), IDR (0)
12				
13				
14				
15				

Allows host to indicate attributes for ranges of logical blocks

- **Each logical block range definition is 16B**
- **Up to 256 range definitions in a command**
- **Range definitions are held in a contiguous buffer that is up to 4KB in size**
- **Buffer is defined by PRP1 and PRP2**

- **Number of Ranges** – number of range definitions associated with the command
- **Integral Dataset for Read (IDR)** – Indicates that dataset (all provided ranges) should be optimized to be read as a single unit. If a potion of the dataset is read, it is expected that all range definitions will be read.
- **Integral Dataset for Write (IDW)** – Indicates that dataset (all provided ranges) should be optimized to be write as a single unit. If a potion of the dataset is written, it is expected that all range definitions will be written.
- **Deallocate (AD)** – Indicates that all provided ranges may be de-allocated

Range Definition

		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
DWord	0	Context Attributes																															
	1	Length in Logical Blocks																															
	2	Starting LBA																															
	3																																

Buffer	Range 0	Context Attributes
		Length in Logical Blocks
		Starting LBA
	Range 1	Context Attributes
		Length in Logical Blocks
		Starting LBA
	Range 2	Context Attributes
		Length in Logical Blocks
		Starting LBA
	Range 3	Context Attributes
		Length in Logical Blocks
		Starting LBA
	Range 4	Context Attributes
		Length in Logical Blocks
		Starting LBA

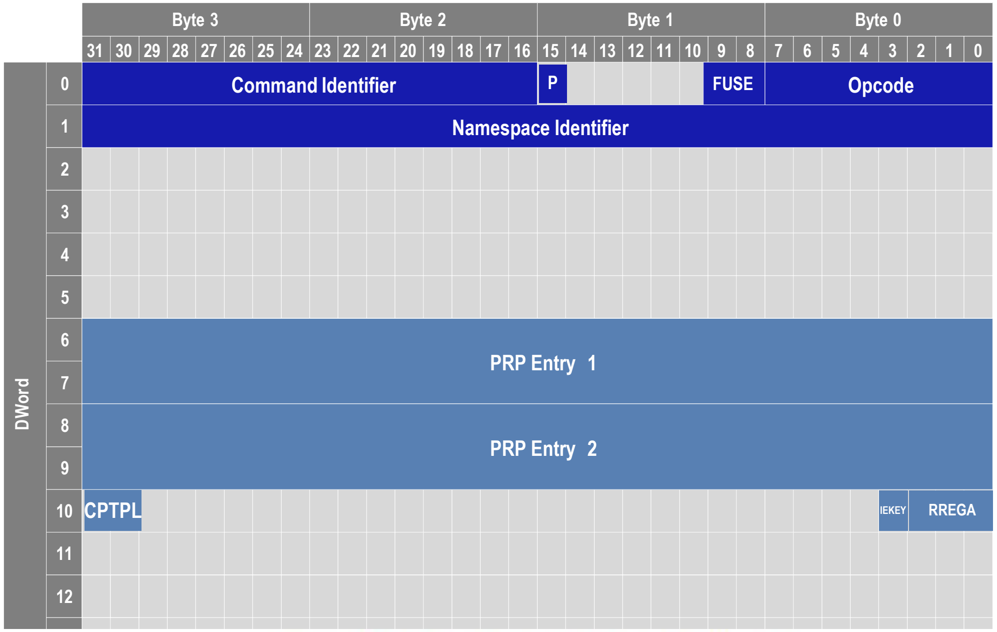
- **Context Attributes** – provides information on how range will be used by host software (described later)
- **Length in Logical Block** – number of logical blocks associated with range defintion
- **Starting LBA** – Range definition logical block starting address

Context Attributes

	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 Access Size																					WP	SW	SR			AL		AF			

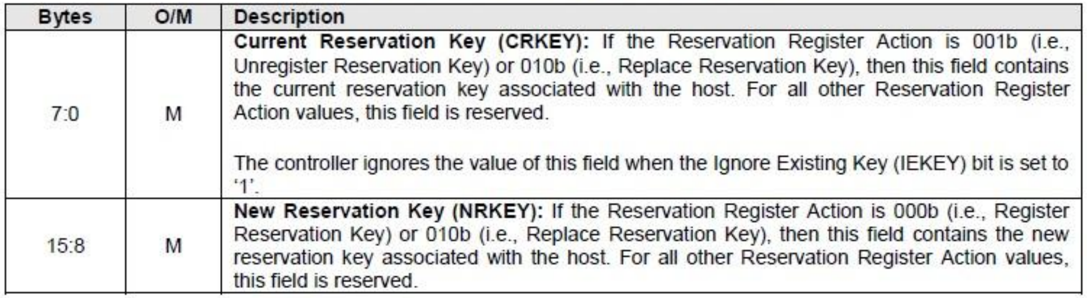
- **Access frequency (AF)**
  - No info provided
  - Typical access
  - Infrequent access
  - Infrequent writes and frequent reads
  - Frequent writes and infrequent reads
  - Frequent writes and frequent reads
- **Access Latency (AL)**
  - No info provided
  - Longer latency acceptable
  - Typical latency
  - Smallest latency possible
- **Sequential Read Range (SR)** – Optimize for sequential reads as a single object
- **Sequential Write Range (SW)** – Optimize for sequential writes as a single object
- **Write Prepare (WP)** – Range is expected to be written in the near future
- **Command Access Size** – Number of logical block that are expected to be accessed in a read or write command in the near future. Zero indicates no information provided

Reservation Register



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		P FUSE	Opcode
1	Namespace Identifier			
2				
3				
4				
5				
6	PRP Entry 1			
7				
8	PRP Entry 2			
9				
10	CPTPL			IEKEY RREGA
11				
12				

Figure 151: Reservation Register Data Structure



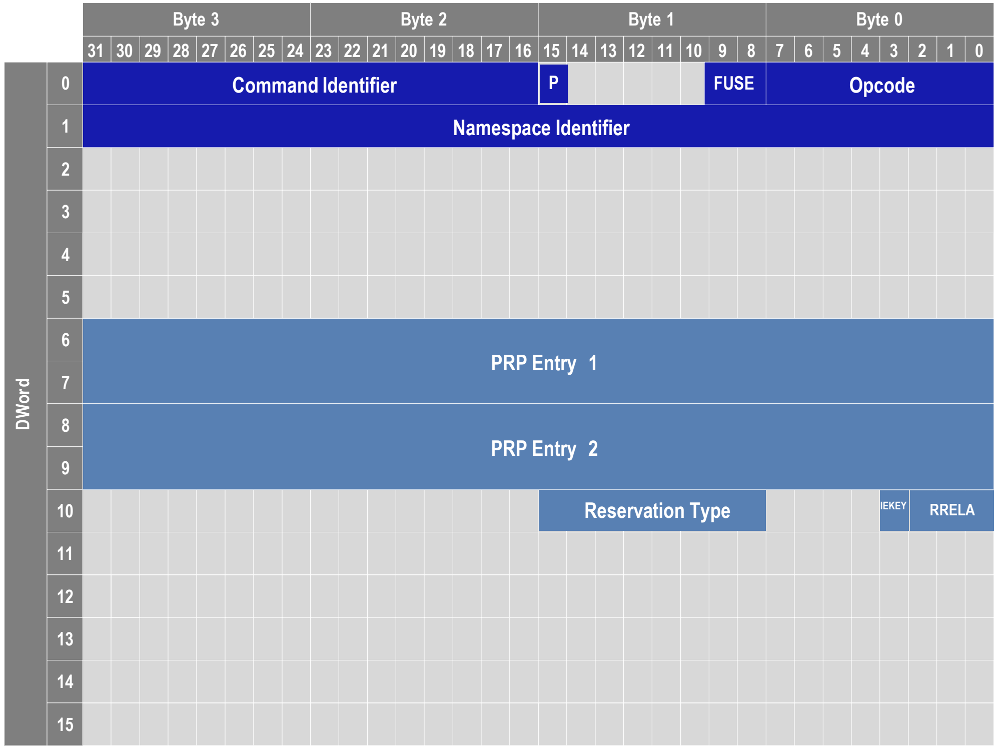
Bytes	O/M	Description
7:0	M	**Current Reservation Key (CRKEY):** If the Reservation Register Action is 001b (i.e., Unregister Reservation Key) or 010b (i.e., Replace Reservation Key), then this field contains the current reservation key associated with the host. For all other Reservation Register Action values, this field is reserved. The controller ignores the value of this field when the Ignore Existing Key (IEKEY) bit is set to '1'.
15:8	M	**New Reservation Key (NRKEY):** If the Reservation Register Action is 000b (i.e., Register Reservation Key) or 010b (i.e., Replace Reservation Key), then this field contains the new reservation key associated with the host. For all other Reservation Register Action values, this field is reserved.

The Reservation Register command is used to register, unregister, or replace a reservation key

- **Change Persist Through Power Loss State (CPTPL):** This field allows the Persist Through Power Loss state associated with the namespace to be modified as a side effect of processing this command
  - 00b No change to PTPL state
  - 10b Set PTPL state to '0'. Reservations are released and registrants are cleared on a power on
  - 11b Set PTPL state to '1'. Reservations and registrants persist across a power loss
- **Ignore Existing Key (IEKEY):** If this bit is set to a '1', then Reservation Register Action (RREGA) field values that use the Current Reservation Key (CRKEY) shall succeed regardless of the value of the Current Reservation Key field in the command (i.e., the current reservation key is not checked)

  Reservation Register Action (RREGA): specifies the action that is performed by the command.
  - 000b Register Reservation Key
  - 001b Unregister Reservation Key
  - 010b Replace Reservation Key

Reservation Release

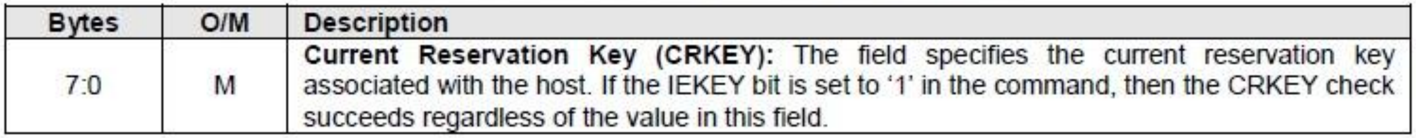


DWord	Byte 3	Byte 2	Byte 1	Byte 0
0	Command Identifier		P / FUSE	Opcode
1	Namespace Identifier			
2				
3				
4				
5				
6	PRP Entry 1			
7				
8	PRP Entry 2			
9				
10			Reservation Type	IEKEY / RRELA
11				
12				
13				
14				
15				

The Reservation Release command is used to release or clear a reservation held on a namespace

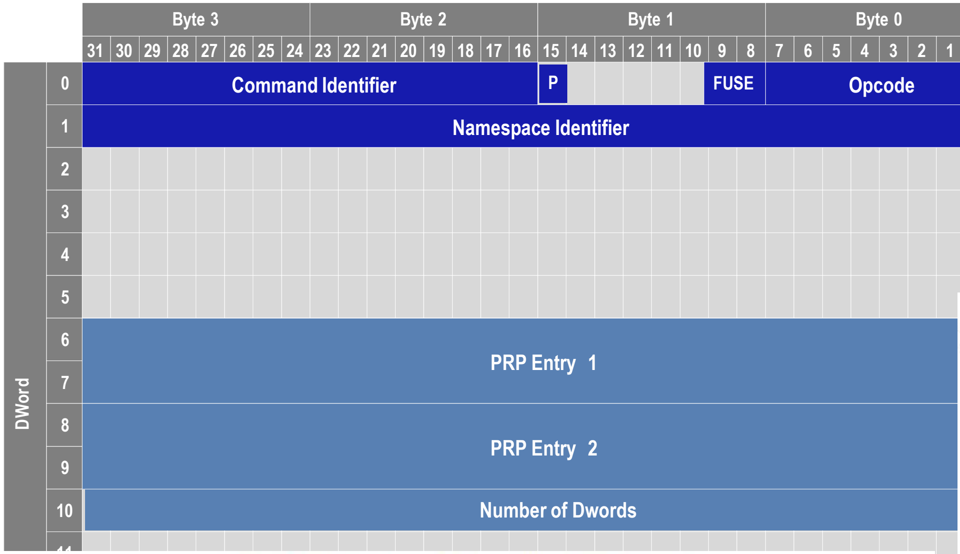
- **Reservation Type (RTYPE**) If the Reservation Release Action is 00b (i.e., Release), then this field specifies the type of reservation that is being released. The reservation type in this field shall match the current reservation type
- **Ignore Existing Key (IEKEY):** If this bit is set to a ‘1’, then the Current Reservation Key (CRKEY) check is disabled and the command succeeds regardless of the CRKEY field value
- **Reservation Release Action (RRELA):** specifies the registration action that is performed by the command.
  - 00b Release
  - 01b Clear

Figure 154: Reservation Release Data Structure



Bytes	O/M	Description
7:0	M	**Current Reservation Key (CRKEY):** The field specifies the current reservation key associated with the host. If the IEKEY bit is set to ‘1’ in the command, then the CRKEY check succeeds regardless of the value in this field.

Reservation Report



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		P FUSE	Opcode
1	Namespace Identifier			
2				
3				
4				
5				
6	PRP Entry 1			
7				
8	PRP Entry 2			
9				
10	Number of Dwords			

The Reservation Report command returns a Reservation Status data structure to host memory that describes the registration and reservation status of a namespace

- **Number of Dwords (NUMD):** specifies the number of Dwords of the Reservation Status data structure to transfer.

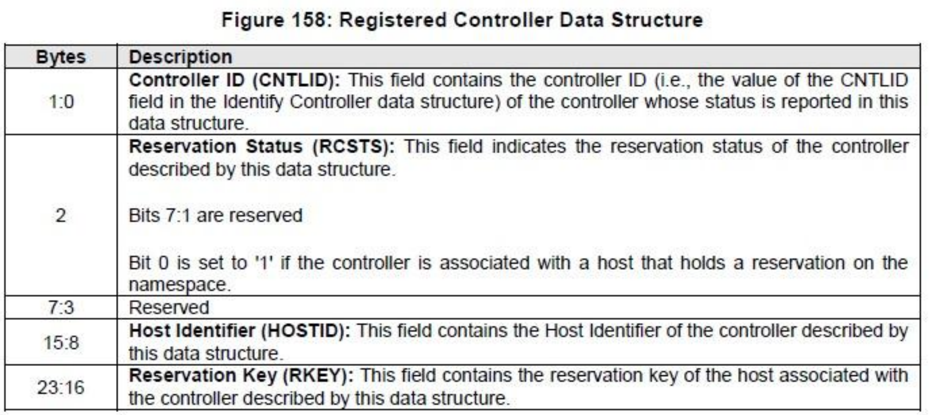


Figure 158: Registered Controller Data Structure

Bytes	Description
1:0	**Controller ID (CNTLID):** This field contains the controller ID (i.e., the value of the CNTLID field in the Identify Controller data structure) of the controller whose status is reported in this data structure.
2	**Reservation Status (RCSTS):** This field indicates the reservation status of the controller described by this data structure. Bits 7:1 are reserved Bit 0 is set to '1' if the controller is associated with a host that holds a reservation on the namespace.
7:3	Reserved
15:8	**Host Identifier (HOSTID):** This field contains the Host Identifier of the controller described by this data structure.
23:16	**Reservation Key (RKEY):** This field contains the reservation key of the host associated with the controller described by this data structure.

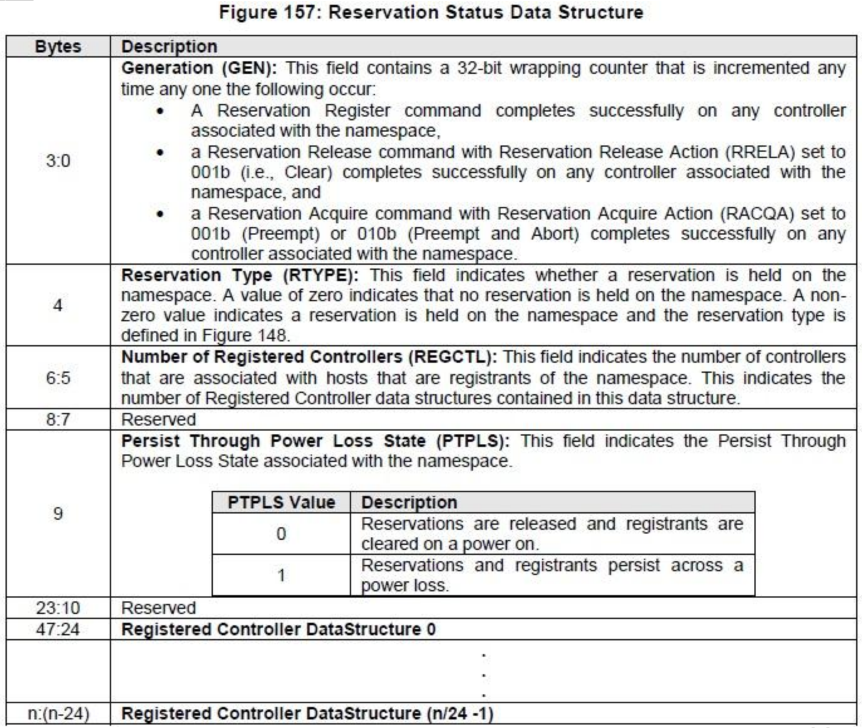
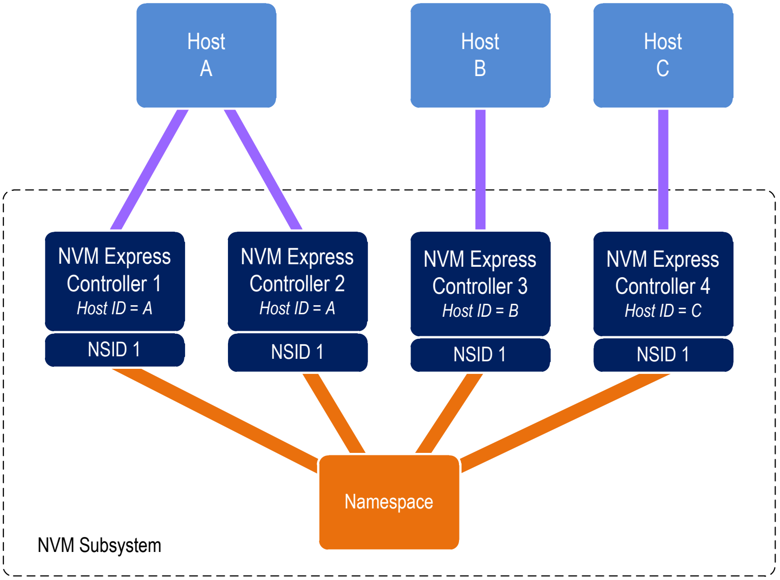


Figure 157: Reservation Status Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>3:0</td><td><b>Generation (GEN):</b> This field contains a 32-bit wrapping counter that is incremented any time any one the following occur:
• A Reservation Register command completes successfully on any controller associated with the namespace,
• a Reservation Release command with Reservation Release Action (RRELA) set to 001b (i.e., Clear) completes successfully on any controller associated with the namespace, and
• a Reservation Acquire command with Reservation Acquire Action (RACQA) set to 001b (Preempt) or 010b (Preempt and Abort) completes successfully on any controller associated with the namespace.</td></tr>
<tr><td>4</td><td><b>Reservation Type (RTYPE):</b> This field indicates whether a reservation is held on the namespace. A value of zero indicates that no reservation is held on the namespace. A non-zero value indicates a reservation is held on the namespace and the reservation type is defined in Figure 148.</td></tr>
<tr><td>6:5</td><td><b>Number of Registered Controllers (REGCTL):</b> This field indicates the number of controllers that are associated with hosts that are registrants of the namespace. This indicates the number of Registered Controller data structures contained in this data structure.</td></tr>
<tr><td>8:7</td><td>Reserved</td></tr>
<tr><td>9</td><td><b>Persist Through Power Loss State (PTPLS):</b> This field indicates the Persist Through Power Loss State associated with the namespace.
<table>
<tr><th>PTPLS Value</th><th>Description</th></tr>
<tr><td>0</td><td>Reservations are released and registrants are cleared on a power on.</td></tr>
<tr><td>1</td><td>Reservations and registrants persist across a power loss.</td></tr>
</table></td></tr>
<tr><td>23:10</td><td>Reserved</td></tr>
<tr><td>47:24</td><td><b>Registered Controller DataStructure 0</b></td></tr>
<tr><td></td><td>.
.
.</td></tr>
<tr><td>n:(n-24)</td><td><b>Registered Controller DataStructure (n/24 -1)</b></td></tr>
</table>

Reservations in Action

- Example: Host A and B have read/write access and host C has read-only access to the shared namespace



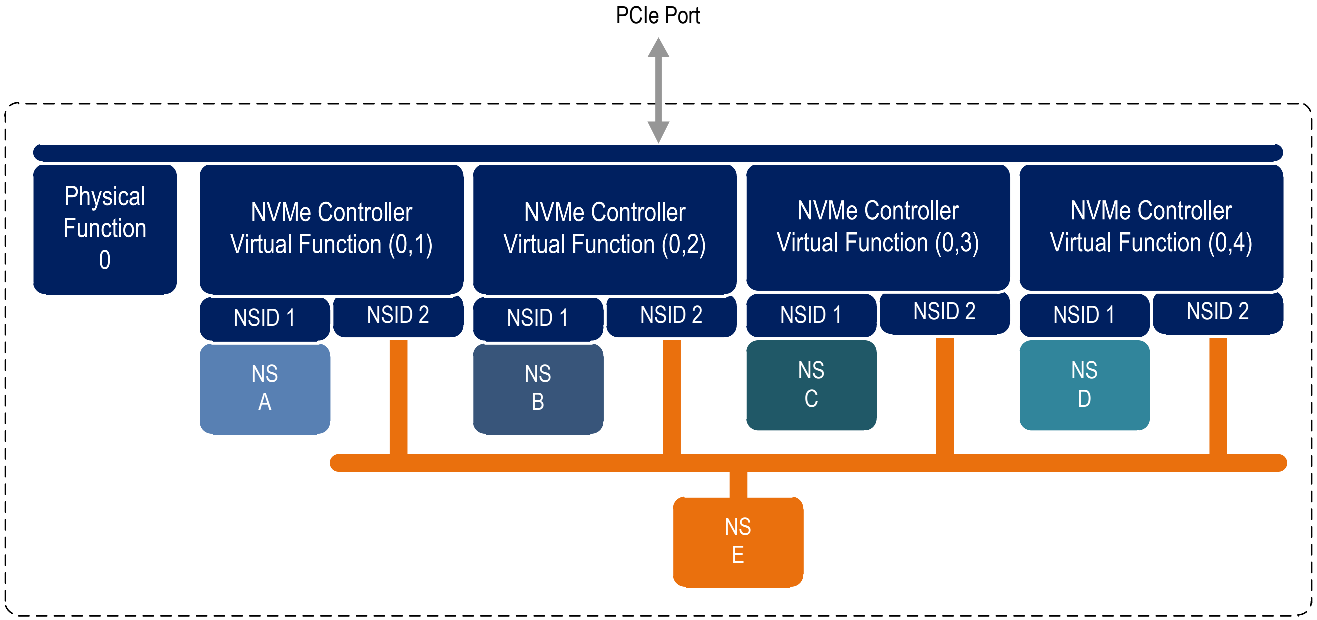
```
HostA-SetFeatures (HostID_A) -> OK
HostB-SetFeatures (HostID_B) -> OK
HostC-SetFeatures (HostID_C) -> OK
…
HostA-Register(NSID,Key_A) -> OK
HostB-Register(NSID,Key_B) -> OK
HostA-AcquireReservation(NSID, Reservation, WriteExclusiveRegistrantsOnly,Key_A) -> OK
HostC-AcquireReservation(NSID, Reservation, WriteExclusiveRegistrantsOnly,Key_C) ->
          Error – Reservation Conflict
…
HostA-Write(NSID)  -> OK
…
HostB-Read(NSID)  -> OK
…
HostB-Write(NSID)  -> OK
…
HostC->Read(NSID) -> OK
HostC->Write(NSID) -> Error – Reservation Conflict
…
HostA-ReleaseReservation(NSID,Key1) -> OK
HostC-Write(NSID) -> OK
…
```

Queue Management

To allocate I/O Submission Queues and I/O Completion Queues, host software follows these steps:

1. Configure the Admin Registers and enable controller (CC.EN=1)
2. Submit a Set Features command for the Number of Queues attribute in order to request the number of I/O Submission Queues and I/O Completion Queues desired. The completion of this Set Features command indicates the number of I/O Submission and Completion Queues allocated.
3. Determine the maximum number of entries supported per queue (CAP.MQES) and whether the queues are required to be physically contiguous (CAP.CQR)
4. Allocate the desired I/O Completion Queues by using the Create I/O Completion Queue command.
5. Allocates the desired I/O Submission Queues by using the Create I/O Submission Queue command.

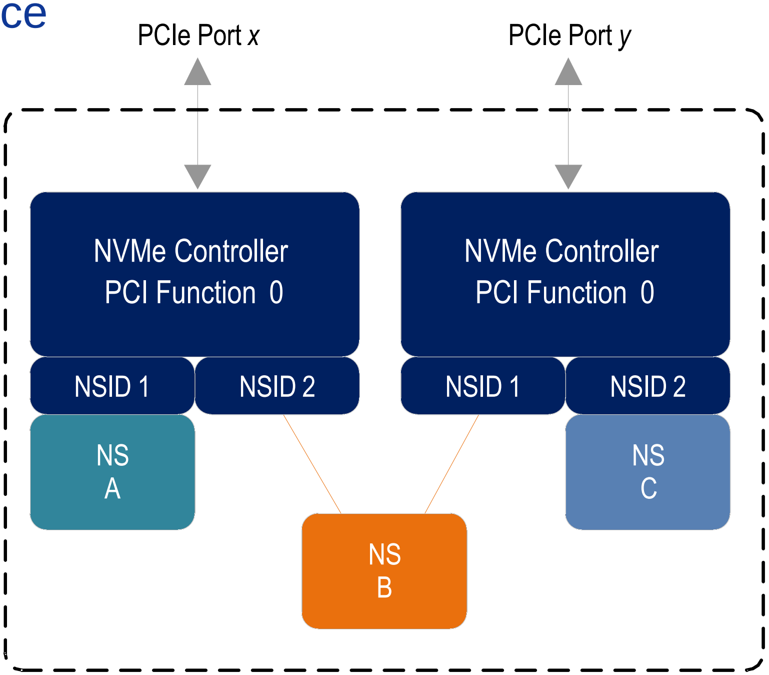
PCI Express SR-IOV



PCIe Port
Physical Function 0
NVMe Controller Virtual Function (0,1)
NVMe Controller Virtual Function (0,2)
NVMe Controller Virtual Function (0,3)
NVMe Controller Virtual Function (0,4)
NSID 1
NSID 2
NSID 1
NSID 2
NSID 1
NSID 2
NSID 1
NSID 2
NS A
NS B
NS C
NS D
NS E

Multi-Path I/O and Namespace Sharing

- An NVMe namespace may be accessed via multiple "paths"
  - SSD with multiple PCI Express* ports
  - SSD behind a PCIe switch to many hosts
- Two hosts accessing the same namespace must coordinate
- NVM Express 1.1 added hooks to enable Enterprise multi-host usage models
  - Globally Unique ID for a namespace
  - Reservation capability



Flash Memory Summit 2013
Santa Clara, CA

Controller Shutdown

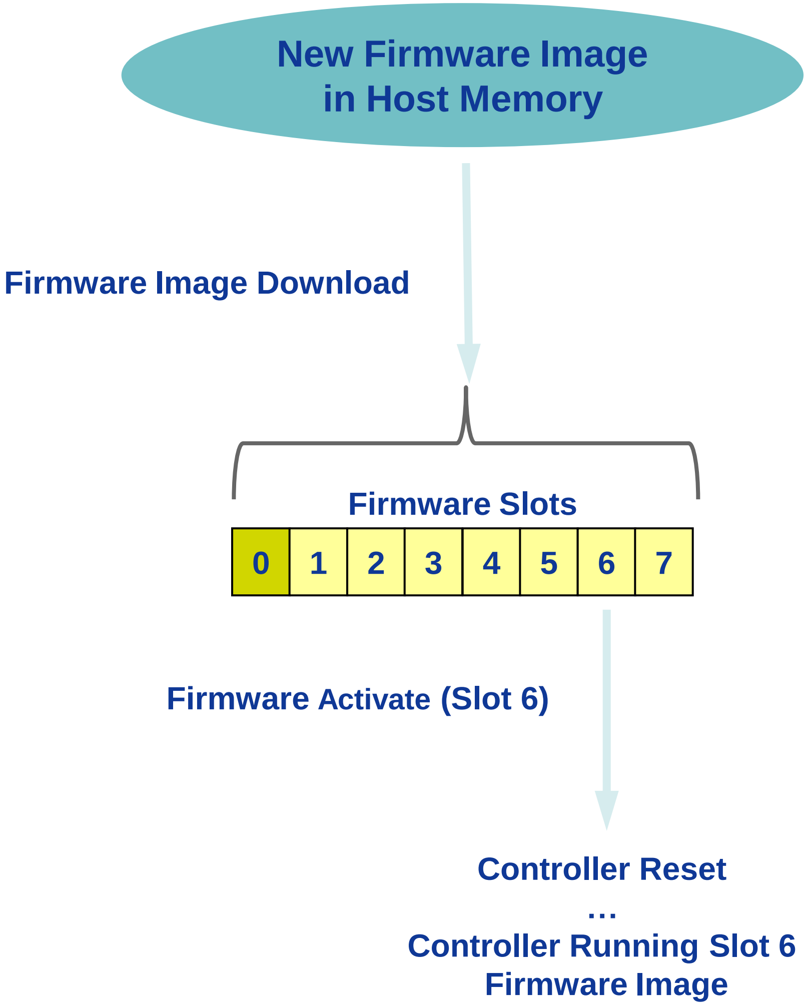
The host performs the following actions in sequence for a normal shutdown:

1. Stop submitting any new I/O commands to the controller and allow any outstanding commands to complete.
2. The host should delete all I/O Submission Queues, using the Delete I/O Submission Queue command.
3. The host should delete all I/O Completion Queues, using the Delete I/O Completion Queue command.
4. The host should set the Shutdown Notification (CC.SHN) field to 01b to indicate a normal shutdown operation. The controller indicates when shutdown processing is completed by updating the Shutdown Status (CSTS.SHST) field to 10b.

The host perform the following actions in sequence for an abrupt shutdown:

1. Stop submitting any new I/O commands to the controller.
2. The host should set the Shutdown Notification (CC.SHN) field to 10b to indicate an abrupt shutdown operation. The controller indicates when shutdown processing is completed by updating the Shutdown Status (CSTS.SHST) field to 10b.

Firmware Update Process



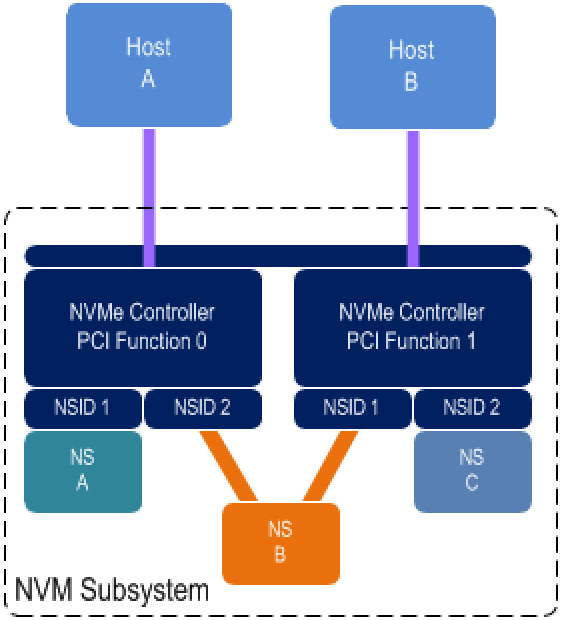
- Firmware slots allows multiple images to be supported
  - Controller supports 1 to 7 slots
  - Slot 0 not a valid slot - reserved
  - Slot 1 may be a read-only firmware image
- Firmware update process
  - Download Firmware Image: controller transfers image from host
  - Activate Firmware:
    - Replace Firmware: controller validates image & applies to selected slot
    - Controller makes selected slot active
  - Firmware update occurs on next reset
- Firmware boot failure
  - Revert to previous active slot or baseline read-only image
  - Host software notified via a Firmware Image Load Error asynchronous event

Resets

- There are five primary controller level reset mechanisms:
  - NVM Subsystem Reset
  - Conventional Reset (PCI Express Hot, Warm, or Cold reset)
  - PCI Express transaction layer Data Link Down status
  - Function Level Reset (PCI reset)
  - Controller Reset (CC.EN transitions from ‘1’ to ‘0’)
- When any of the above resets occur, the following actions are performed:
  - All I/O Submission and Completion Queues are deleted.
  - All outstanding Admin and I/O commands shall be processed as aborted by host software.
  - The controller is brought to an Idle state = CSTS.RDY is cleared to ‘0’.
  - 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 and internal controller state are reset.
- In all cases except a Controller Reset, the PCI register space is reset as defined by the PCI Express base specification.

NVM Subsystem Reset

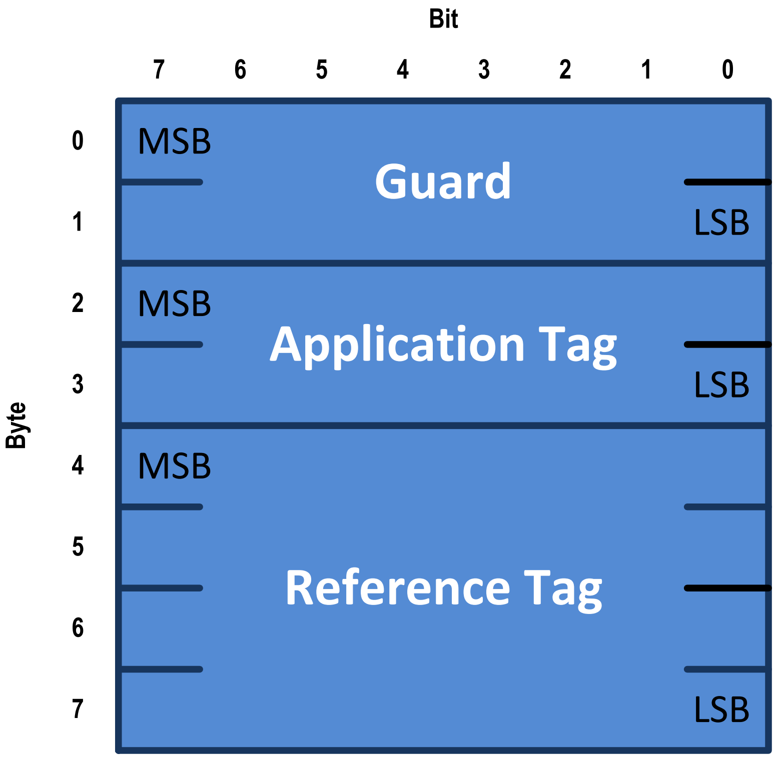
- If an NVM Subsystem Reset occurs, the entire NVM subsystem is reset. This includes the initiation of a Controller Level Reset on all controllers that make up the NVM subsystem and a transition to the Detect LTSSM state by all PCI Express ports of the NVM subsystem.
- An NVM Subsystem Reset is initiated when:
  - Power is applied to the NVM subsystem,
  - A value of 4E564D65h ("NVMe") is written to the NSSR.NSSRC field, or
  - A vendor specific event occurs.
- To perform an NVM Subsystem Reset, write the value "NVMe" to the register



Bit	Type	Reset	Description
31:00	RW	0h	**NVM Subsystem Reset Control (NSSRC):** A write of the value 4E564D65h ("NVMe") to this field initiates an NVM Subsystem Reset. A write of any other value has no functional effect on the operation of the NVM subsystem. This field shall return the value 0h when read.

NVMe = NVM Express

Data Protection



Data protection information associated with each sector

Same format as DIF / DIX

- Guard field
  - CRC-16 as defined by T10 DIF
    - IP Checksum not supported
- Application tag field
  - Same definition as T10 DIF
  - May be used to disable checking of protection information (i.e., 0xFFFF)
  - Generally opaque data not interpreted by controller
- Reference tag field
  - Same definition as T10 DIF
  - May be used to disable checking of protection info (i.e., 0xFFFF_FFFF)
  - Incrementing value associated with sector address or value provided as part of command