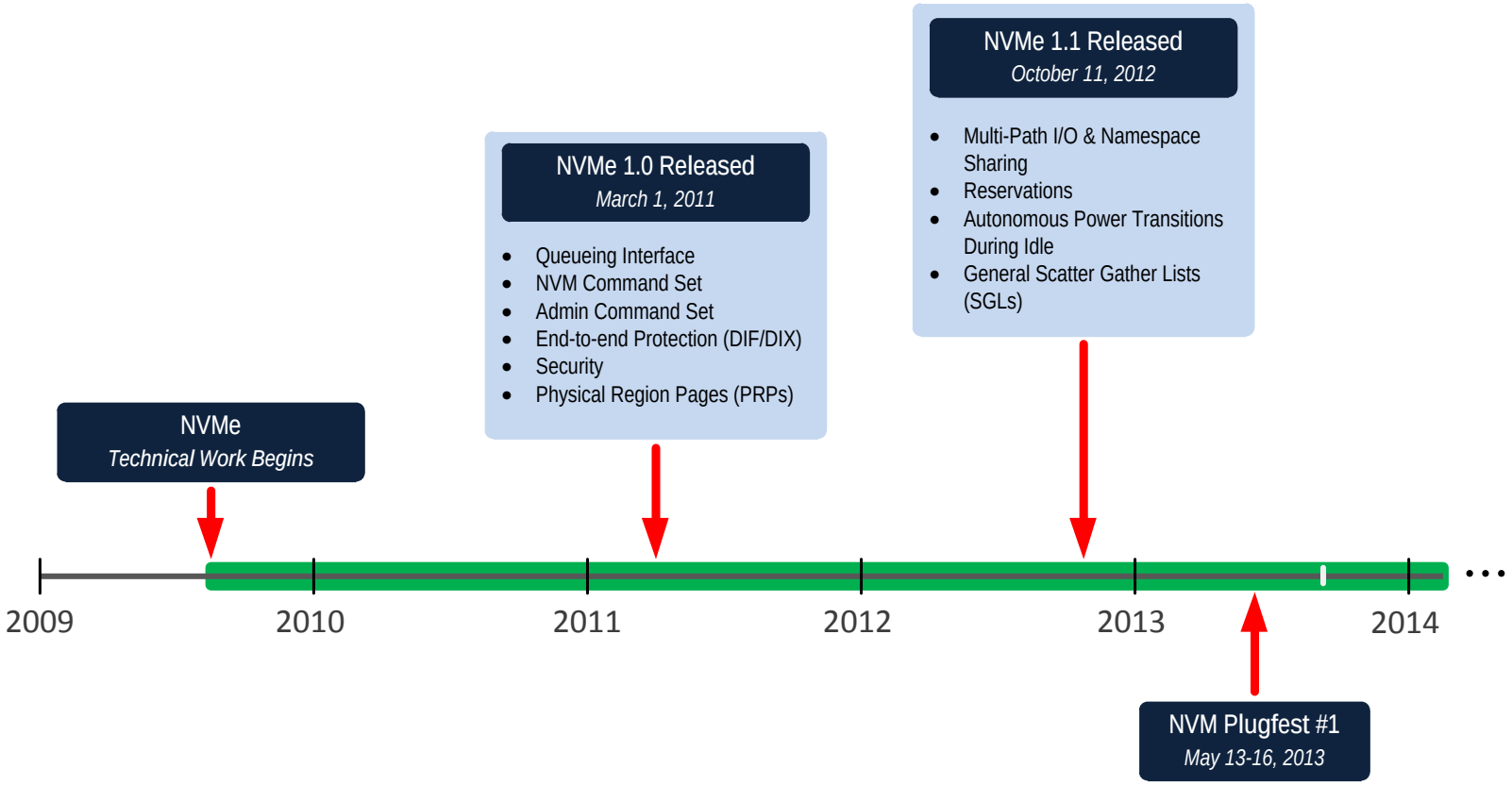




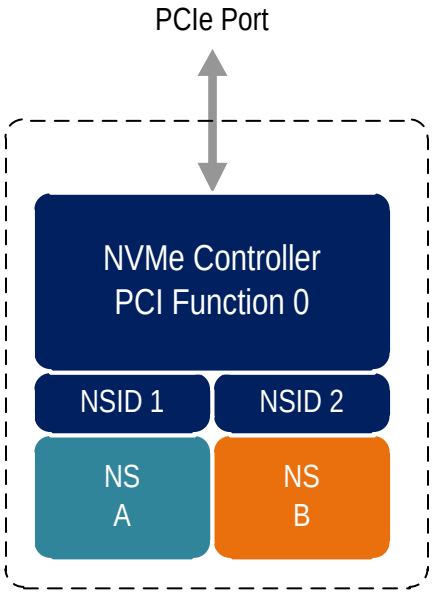
# What's New in NVMe 1.1 and Future Directions

Peter Onufryk
Sr. Director, Product Development
PMC-Sierra

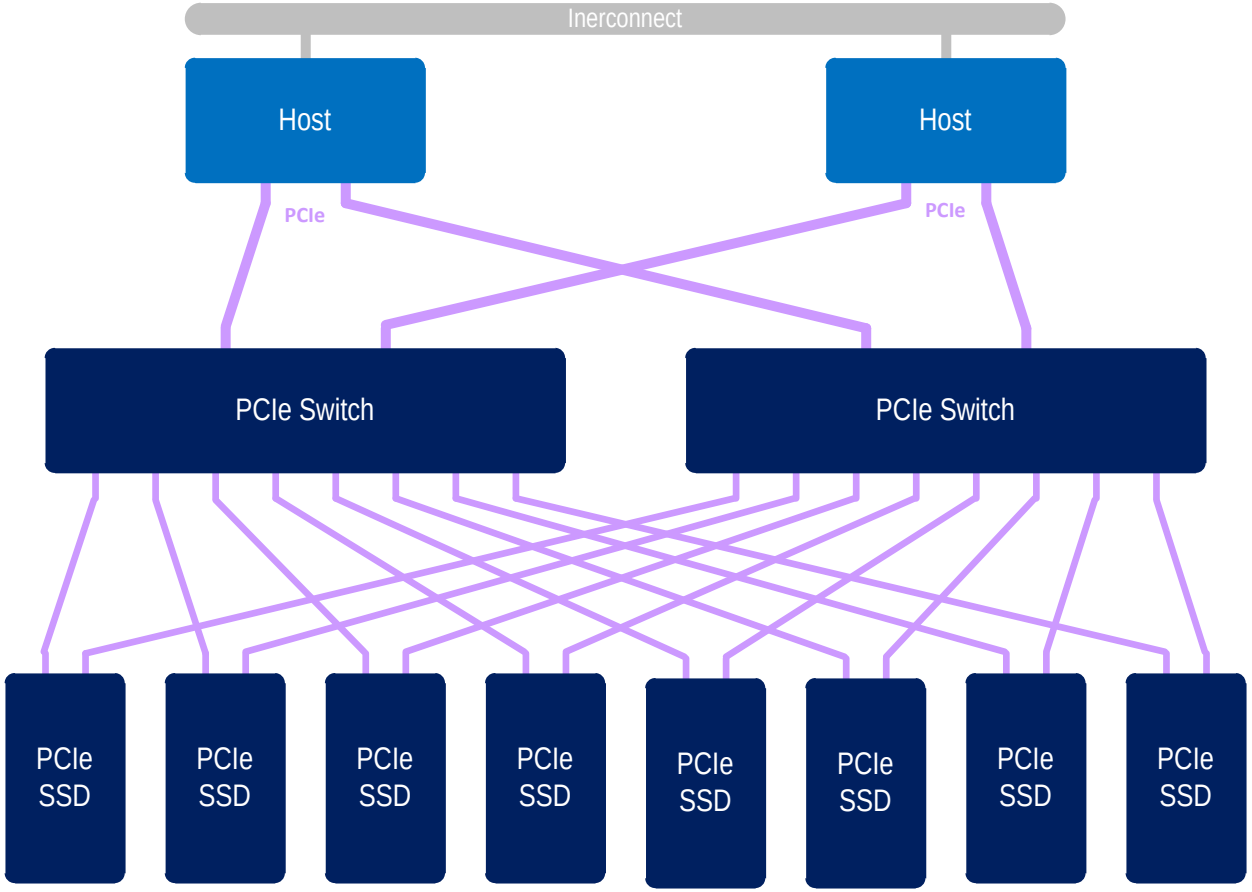
# NVMe Development Timeline



# Architectural Model of NVMe Controller

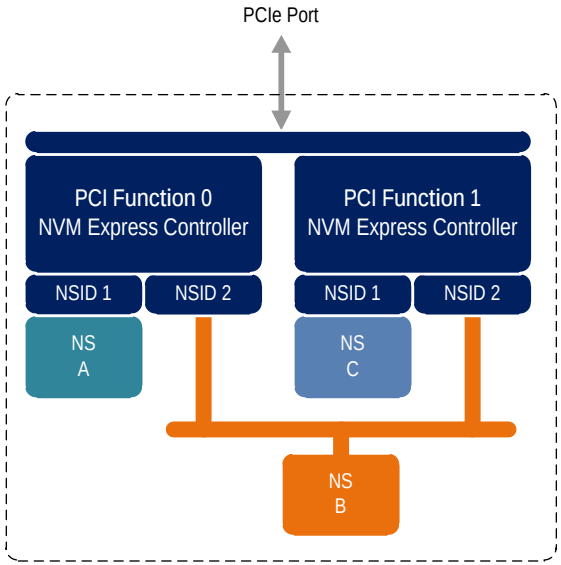


# PCIe Multi-Path Usage Model

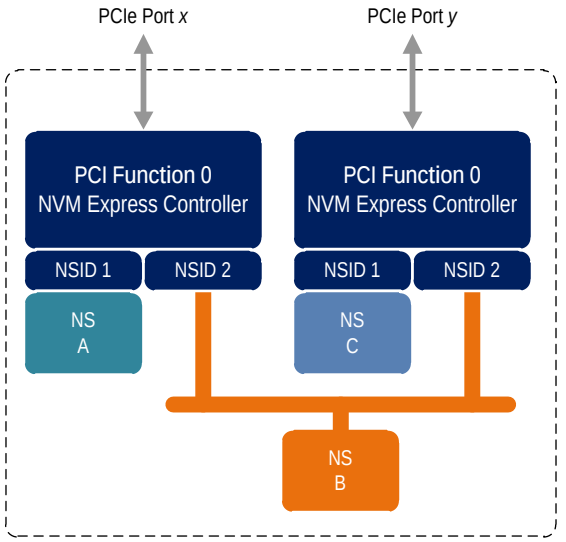


# Multi-Path I/O and Namespace Sharing

- **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

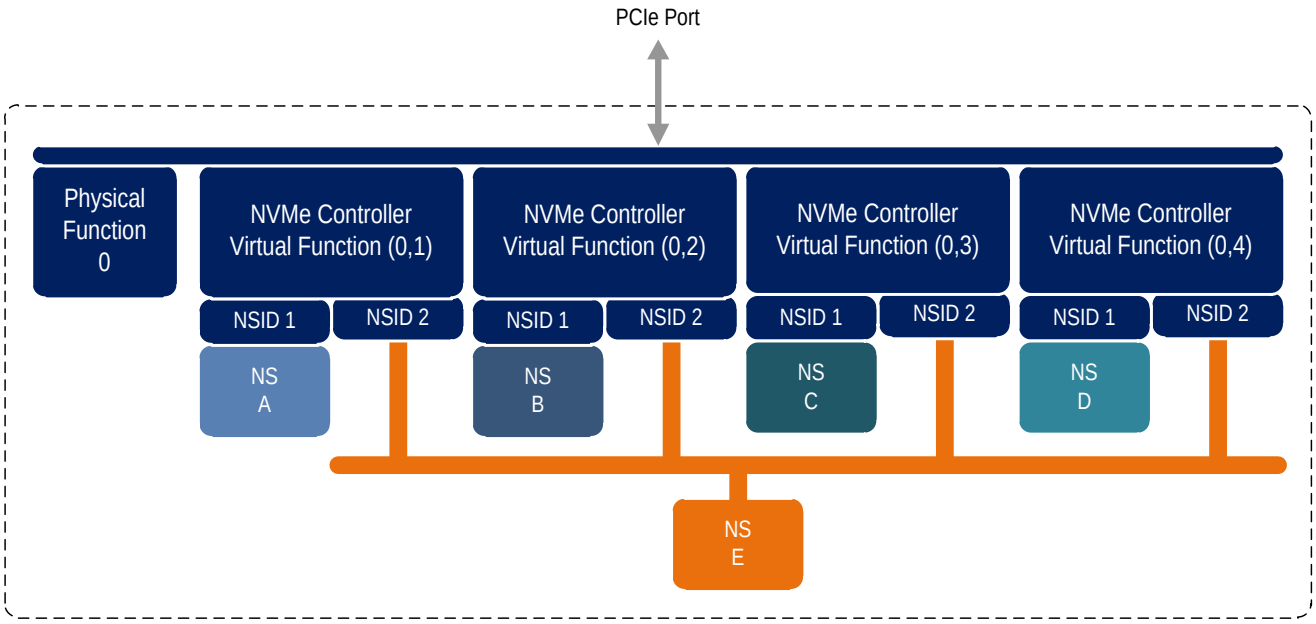



**NVM Subsystem with Two Controllers and One Port**




**NVM Subsystem with Two Controllers and Two Ports**

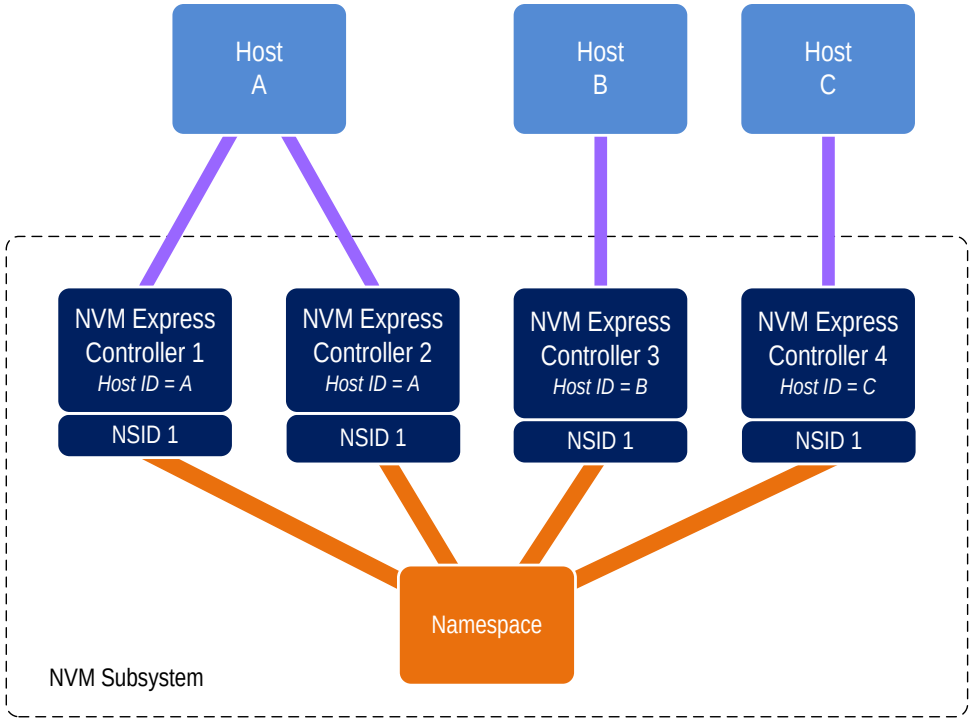
# PCI Express SR-IOV



# Reservation Overview

- Reservations allow two or more hosts to provide coordinate 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
- Reservations are used to restrict access to a 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) preserves reservation properties across all controllers associated with same host

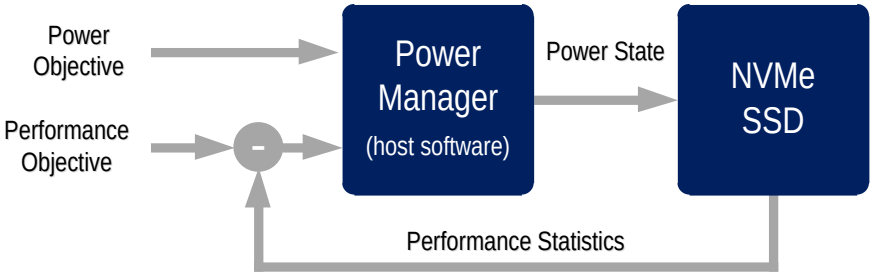
# New NVM Reservation Commands

| NVM I/O Command | Operation |
|---|---|
| Reservation Register | • Register a reservation key<br>• Unregister a reservation key<br>• Replace a reservation key |
| Reservation Acquire | • Acquire a reservation on a namespace<br>• Preempt reservation held on a namespace<br>• Preempt and abort a reservation held on a namespace |
| Reservation Release | • Release a reservation held on a namespace<br>• Clear a reservation held on a namespace |
| Reservation Report | • Retrieve reservation status data structure<br>Type of reservation held on the namespace (if any)<br>Persist through power loss state<br>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 |

# 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 | - | - | - | - |

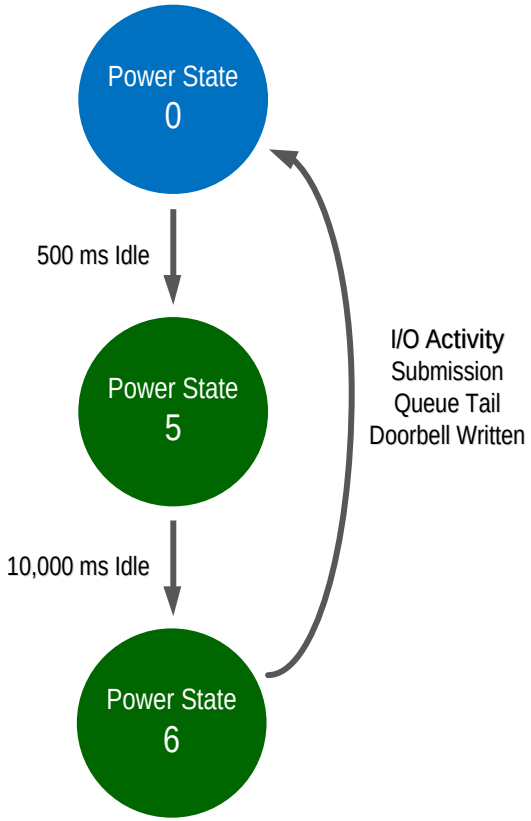
# Autonomous Power State Transitions

**Power State Descriptor Table**

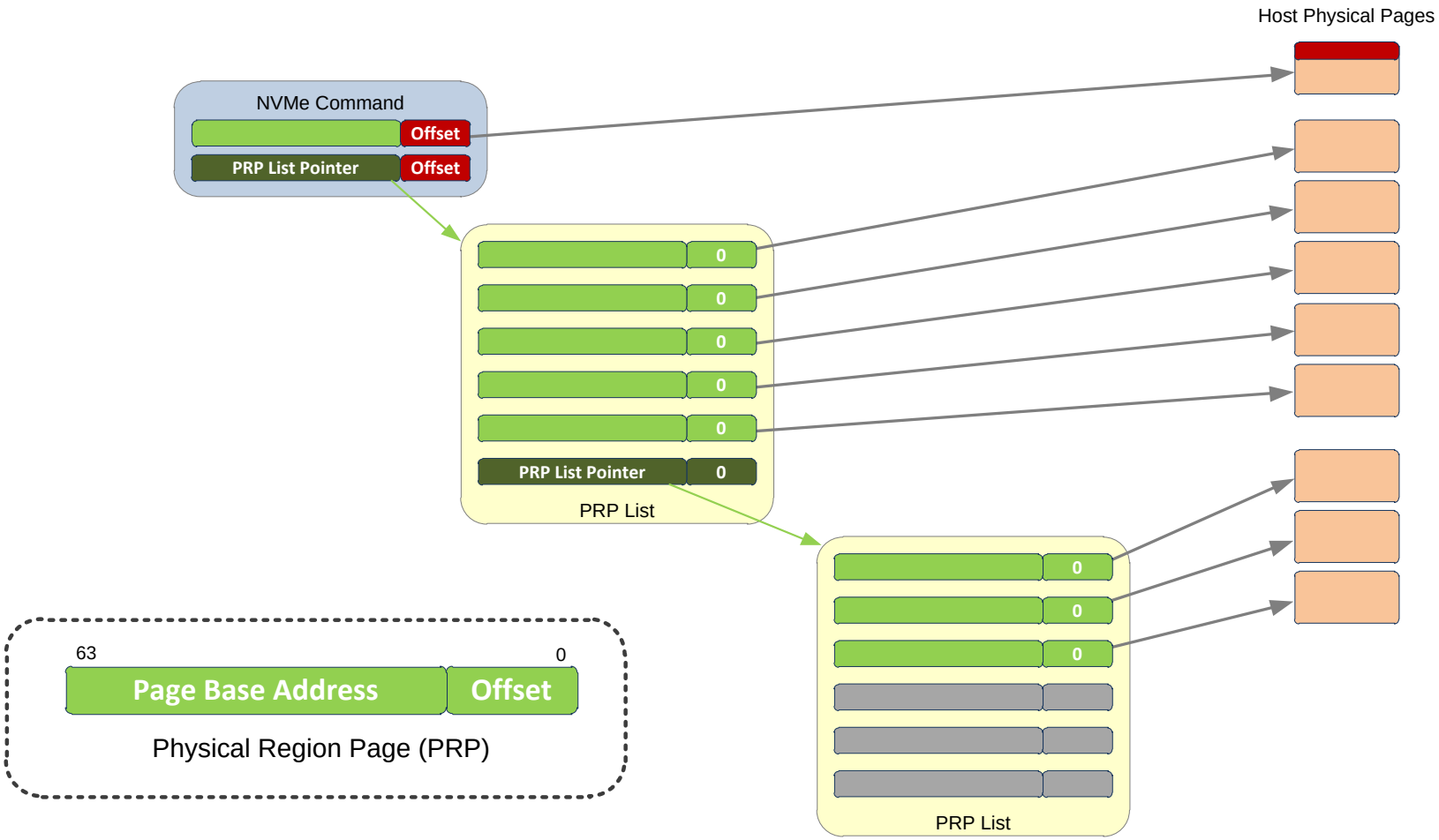
| Power State | Maximum Power | Operational State | Entry Latency | Exit Latency |
|---|---|---|---|---|
| 0 | 25 W | Yes | 5 µs | 5 µs |
| 1 | 18 W | Yes | 5 µs | 7 µs |
| 2 | 18 W | Yes | 5 µs | 8 µs |
| 3 | 15 W | Yes | 20 µs | 15 µs |
| 4 | 7 W | Yes | 20 µs | 30 µs |
| 5 | 1 W | No | 100 mS | 50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

**Autonomous Power State Transition Table**

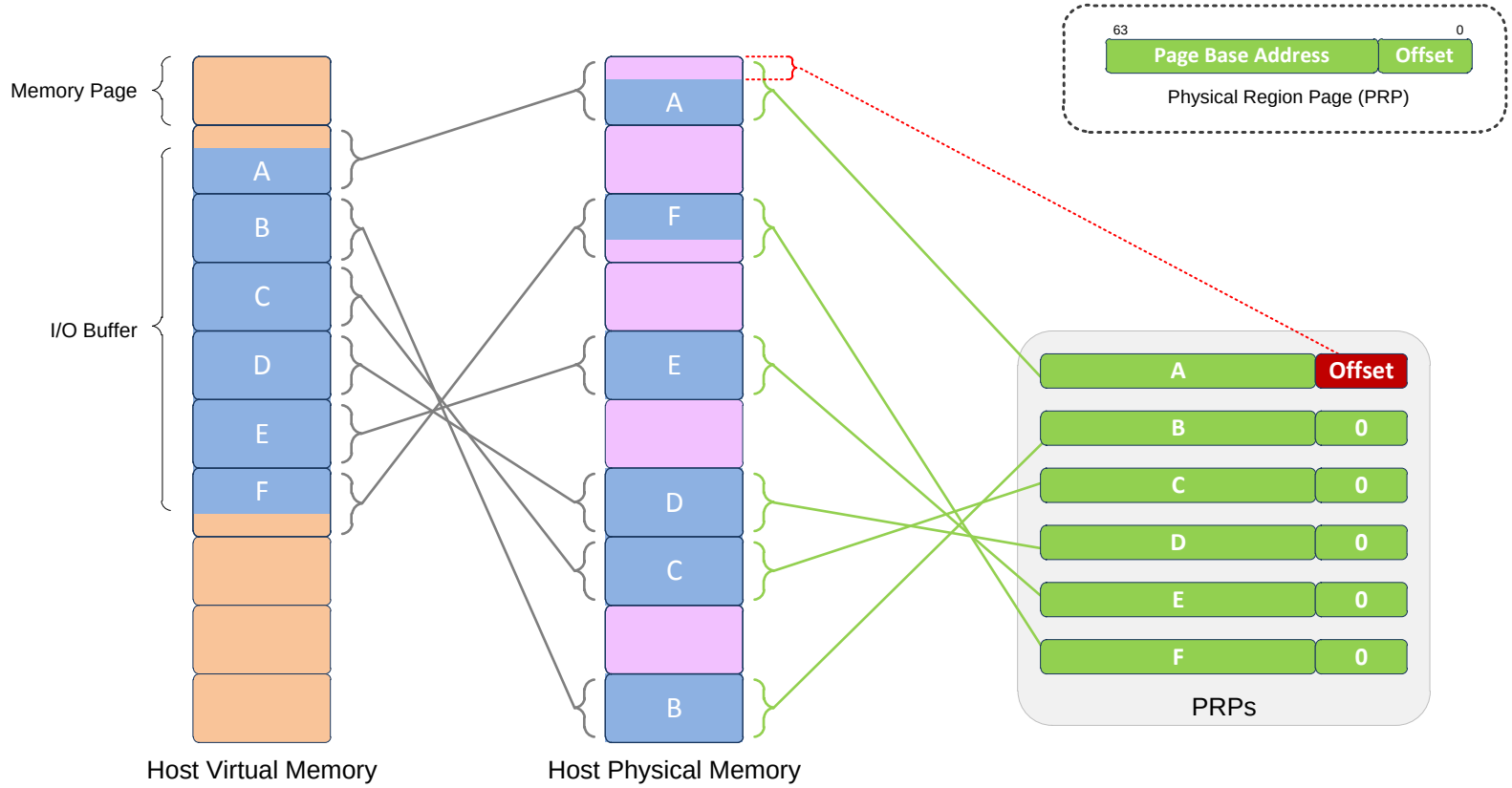
| Idle Time Prior to Transition | Idle Transition Power State |
|---|---|
| 500 ms | 5 |
| 500 ms | 5 |
| 500 ms | 5 |
| 500 ms | 5 |
| 500 ms | 5 |
| 10,000 ms | 6 |
| - | - |



# Physical Region Pages (PRPs)



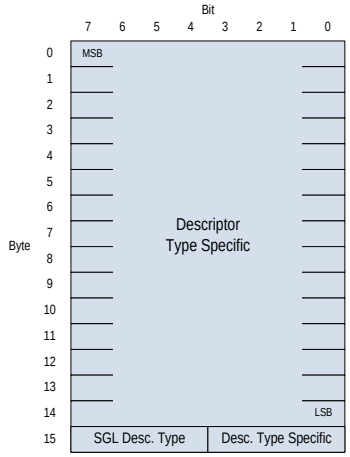
# Why PRPs?




Fixed Size PRPs Accelerate Out of Order Data Delivery

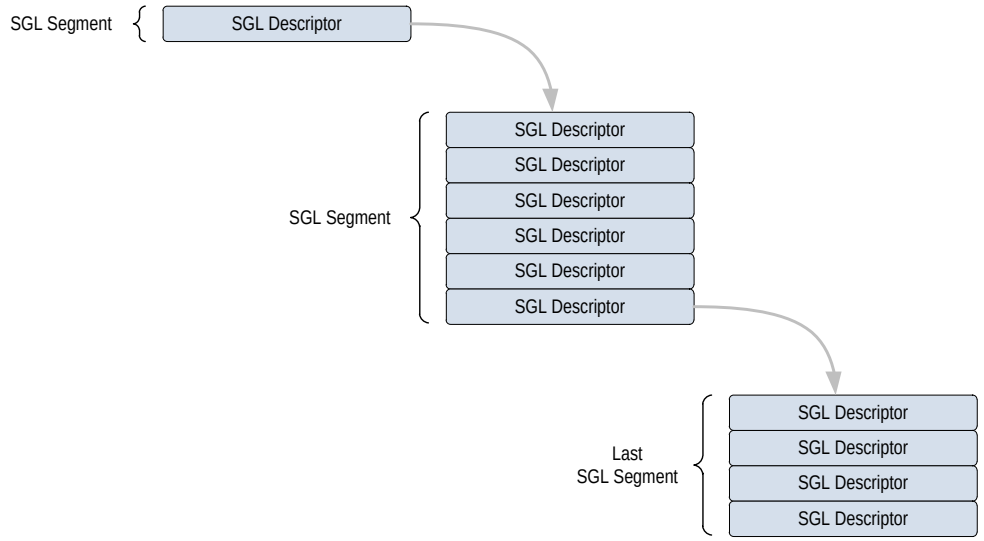
# Scatter Gather List (SGLs)

### SGL Descriptor



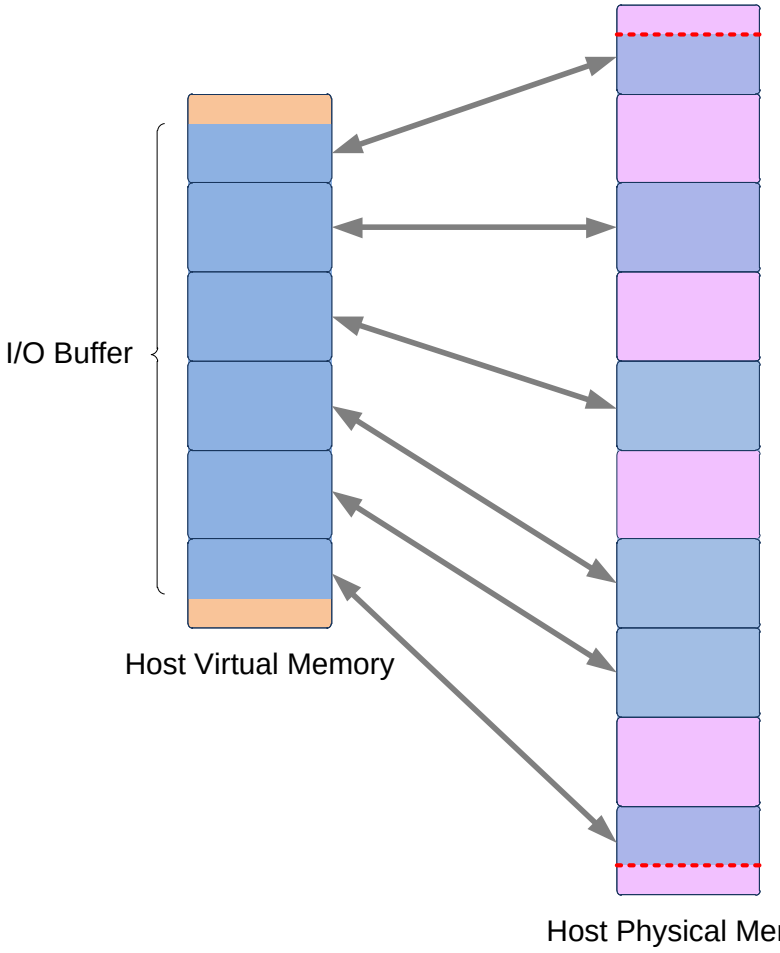

| Code | SGL Descriptor Type |
|---|---|
| 0h | SGL Data Block |
| 1h | SGL Bit Bucket |
| 2h | SGL Segment |
| 3h | SGL Last Segment |
| 4h - Eh | Reserved |
| Fh | Vendor Specific |

### SGL List

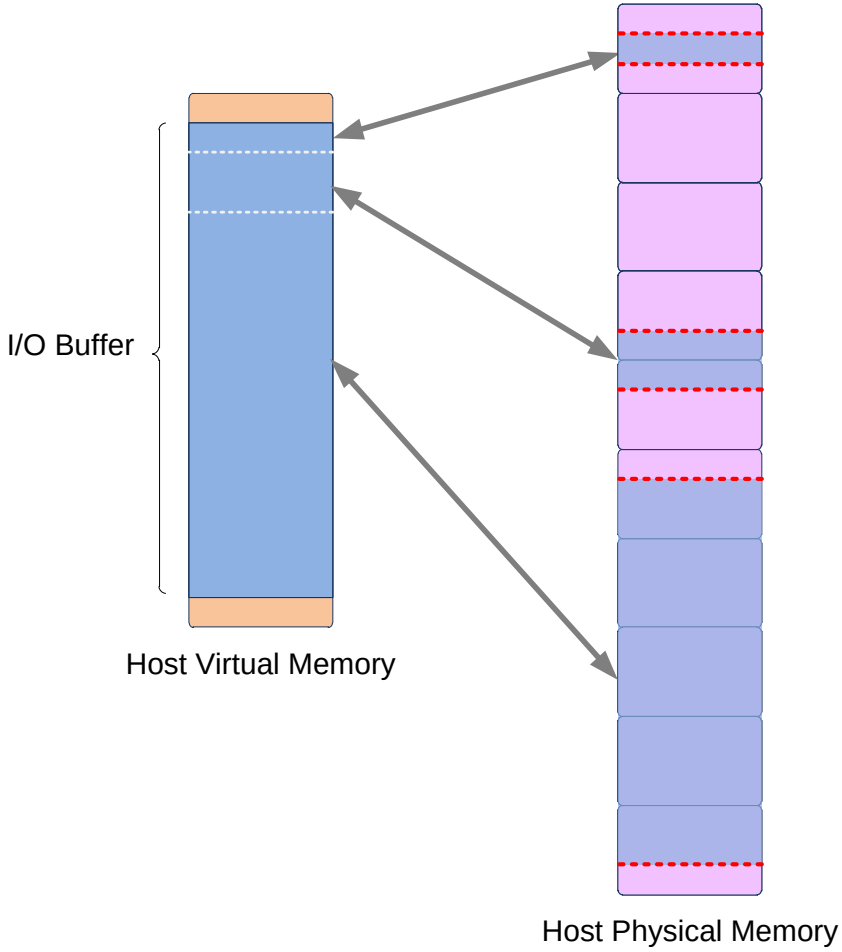



SGLs Enable Arbitrary Data Transfer Size and Byte Alignment

# Comparing SGLs with PRPs

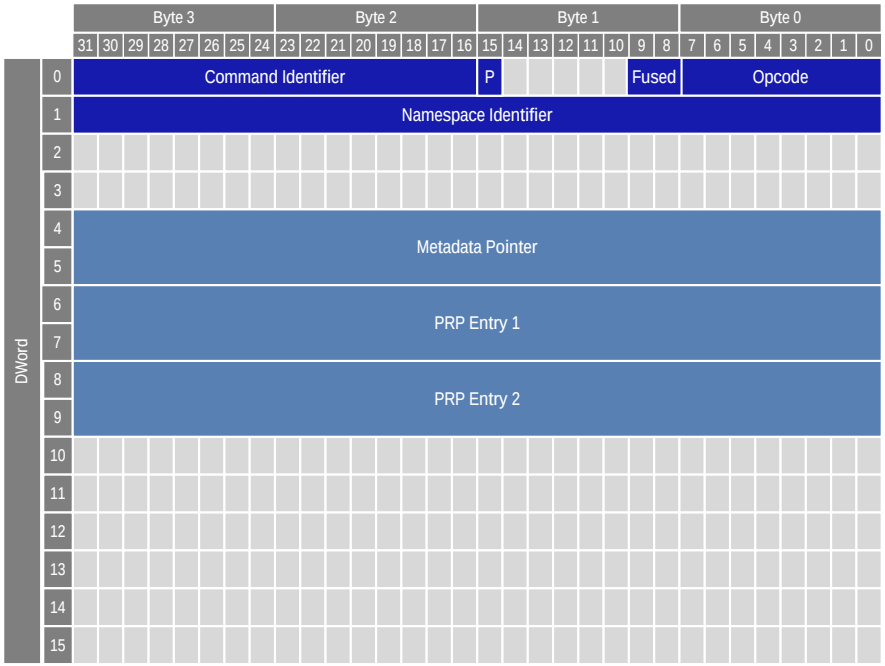



**PRP Data Transfer**



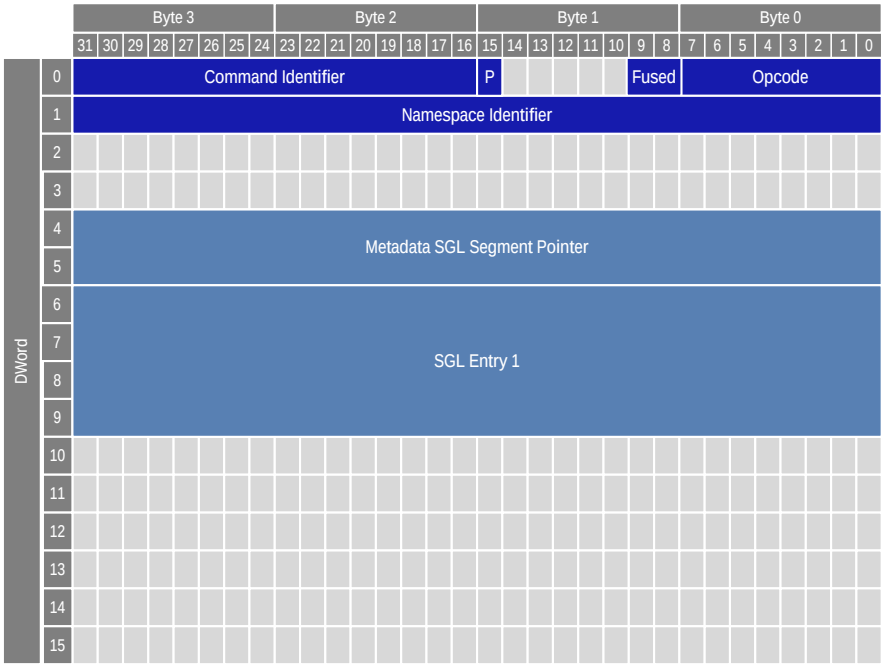

**SGL Data Transfer**

# Command Format (PRP)




- **Opcode** - Command to execute
- **Fused** – Indicates two commands should be executed as atomic unit
- **P** - Use PRPs or SGLs for data transfer
- **Command Identifier** - Unique ID associated with command
- **Namespace Identifier** - Namespace on which command operates
- **Metadata Pointer** – Pointer to contiguous buffer containing metadata in "DIX" mode
- **PRP Entry 1 & 2** – PRP or PRP list

# Command Format (SGL)



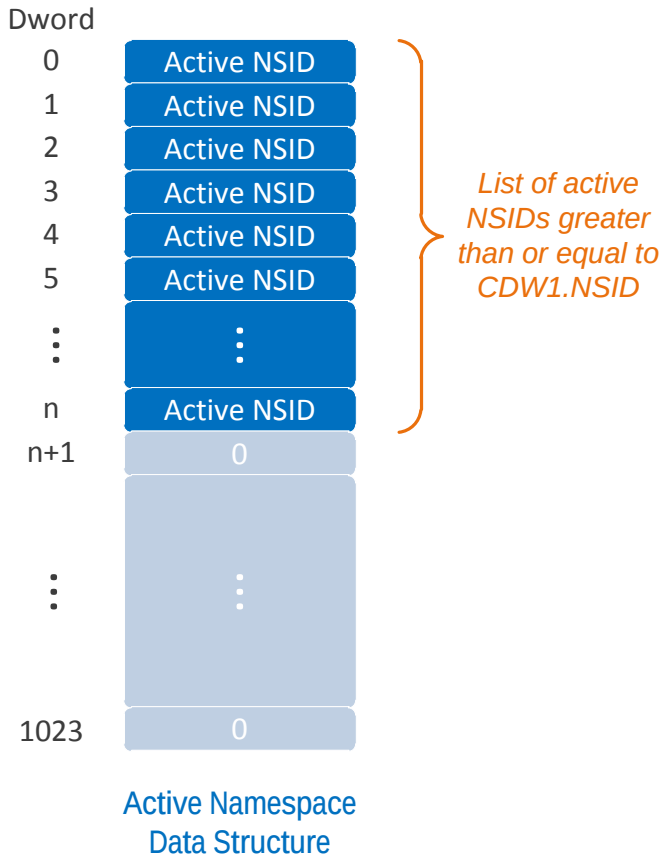

- **Opcode** - Command to execute
- **Fused** – Indicates two simpler commands should be executed as atomic unit
- **P** - Use PRPs or SGLs for data transfer
- **Command Identifier** - Unique ID associated with command
- **Namespace Identifier** - Namespace on which command operates
- **Metadata SGL Segment Pointer** – Pointer to metadata SGL segment in "DIX" mode
- **SGL Entry 1** – First SGL segment associated with data transfer

# Active Namespace Reporting

**Identify**

*Admin Command*

| Identify Controller Data Structure | Identify Namespace Data Structure | Active Namespace Data Structure |
|---|---|---|
| *Return 4KB Identify Controller Data Structure* | *Return 4KB Identify Namespace Data Structure for Namespace Specified in CDW1.NSID* | *Return 4KB Active Namespace Data Starting at Namespace Specified in CDW1.NSID* |



# Other New Features

- Write Zeros Command
  - Ability to set a contiguous range of LBAs to zero
- Subsystem Reset
  - Optional capability to reset entire subsystem
  - Ability to indicate that firmware activation requires subsystem reset
- Persistent Features Across Power States
  - Ability to set the persistence of feature values across power states and resets
- Atomic Compare and Write Unit
  - Atomic write size used by a controller for a compare and write fused command

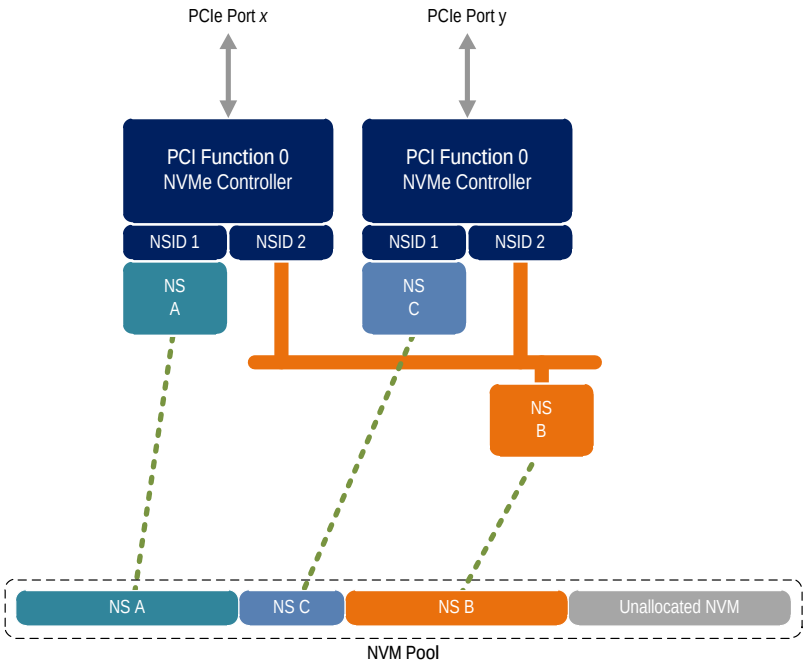
# NVMe 1.1 New Commands

| Admin Commands |
| --- |
| Create I/O Submission Queue |
| Delete I/O Submission Queue |
| Create I/O Completion Queue |
| Delete I/O Completion Queue |
| Get Log Page |
| Identify |
| Abort |
| Set Features |
| Get Features |
| Asynchronous Event Request |
| *Firmware Activate (optional)* |
| *Firmware Image Download (optional)* |

| NVM Admin Commands |
| --- |
| *Format NVM (optional)* |
| *Security Send (optional)* |
| *Security Receive (optional)* |

| NVM I/O Commands |
| --- |
| Read |
| Write |
| Flush |
| *Write Uncorrectable (optional)* |
| *Compare (optional)* |
| *Dataset Management (optional)* |
| ***Write Zeros (optional)*** |
| ***Reservation Register (optional)*** |
| ***Reservation Report (optional)*** |
| ***Reservation Acquire (optional)*** |
| ***Reservation Release (optional)*** |

# Future Direction
## Namespace Management




- Ability to create, resize (larger or small), and delete a namespace
- Ability to attach or detach a namespace to/from a specific controller in the NVM subsystem
- Namespace and NVM pool status reporting
  - What namespaces exist in the NVM subsystem?
  - How big is the NVM pool?
  - How much unallocated space is there in the NVM pool?
  - How much space does a namespace occupy?

# Future Direction
## Namespace Inventory Notice Event

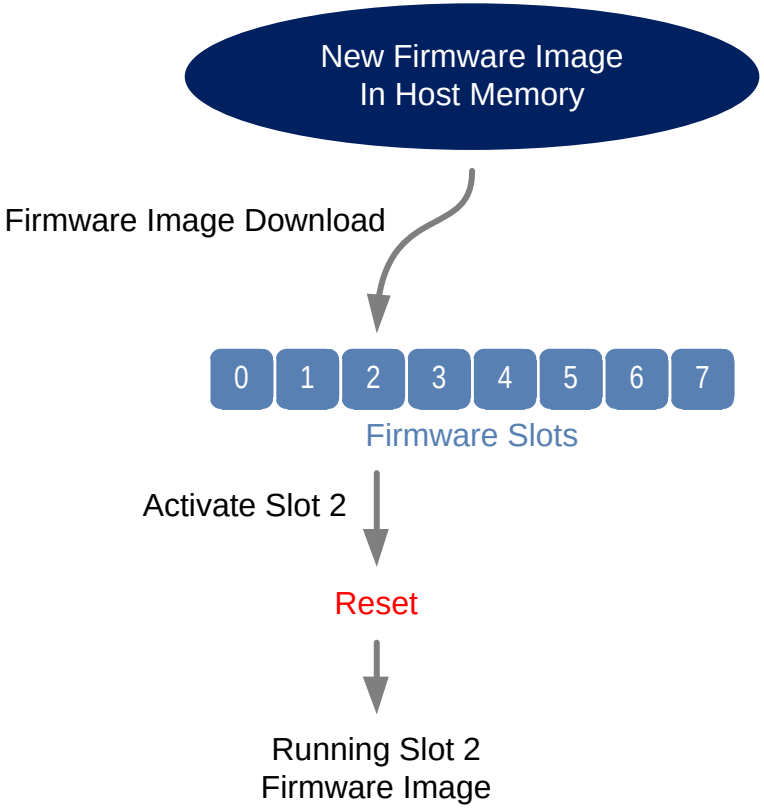
**Figure 84: Identify – Identify Namespace Data Structure, NVM Command Set Specific**

| 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.<br><br>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.<br><br>A value of 0h for the Namespace Capacity indicates that the namespace ID is an inactive namespace ID.<br><br>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.<br><br>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.<br><br>Bits 7:1 are reserved.<br><br>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. |
| 25 | M | **Number of LBA Formats (NLBAF):** This field defines the number of supported LBA data 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.<br><br>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. |

...

- Optionally generate asynchronous event when certain fields in the Identify Namespace data structure change
  - Log page indicates which namespaces are affected
- May be used by host to determine when a namespace change has occurred

# Future Direction
# Firmware Activation Without Reset




- Current firmware update process
  - Download firmware image to firmware slot
  - Activate firmware image
  - Perform reset to cause new firmware image to run
    - Controller reset
    - PCIe conventional reset
    - Subsystem reset
- Enhance firmware update process to allow new firmware image to run without a reset

# *Future Direction*
## Other Enhancements

- More power management enhancements
  - Power state performance and transitional energy
  - Runtime power removal
- SGL enhancements
  - Metadata optimization
- Enhanced error reporting

# Summary

- NVNe 1.1 adds enhancements for client and enterprise applications
  - Multi-Path I/O & Namespace Sharing
  - Reservations
  - Autonomous Power State Transitions During Idle
  - General SGLs
- NVMe 1.1 enhancements maintain backward compatibility
  - Old drivers work with new controllers
  - New drivers work with old controllers
- Future enhancements are planned for NVMe
- NVMe continues to maintain core philosophy of simplicity and efficiency