# NVM Express®

# Computational Programs Command Set Specification

**Revision 1.0**

**December 20th, 2023**

*Please send comments to info@nvmexpress.org*

NVM Express® Computational Programs Command Set is available for download at http://nvmexpress.org. The NVM Express Computational Programs Command Set Specification, Revision 1.0 incorporates TP4091.

SPECIFICATION DISCLAIMER

## Table of Contents

## Table of Figures

# 1   Introduction

## 1.1   Overview

The NVM Express® (NVMe®) Base Specification defines an interface for host software to communicate with a non-volatile memory (NVM) subsystems over a variety of memory-based transports and message-based transports.

This document defines a specific NVMe I/O Command Set, the Computational Programs Command Set, which extends the NVM Express Base Specification.

## 1.2   Scope

Figure 1 shows the relationship of the Computational Programs Command Set Specification to other specifications within the NVMe Family of Specifications.

**Figure 1: NVMe Family of Specifications**
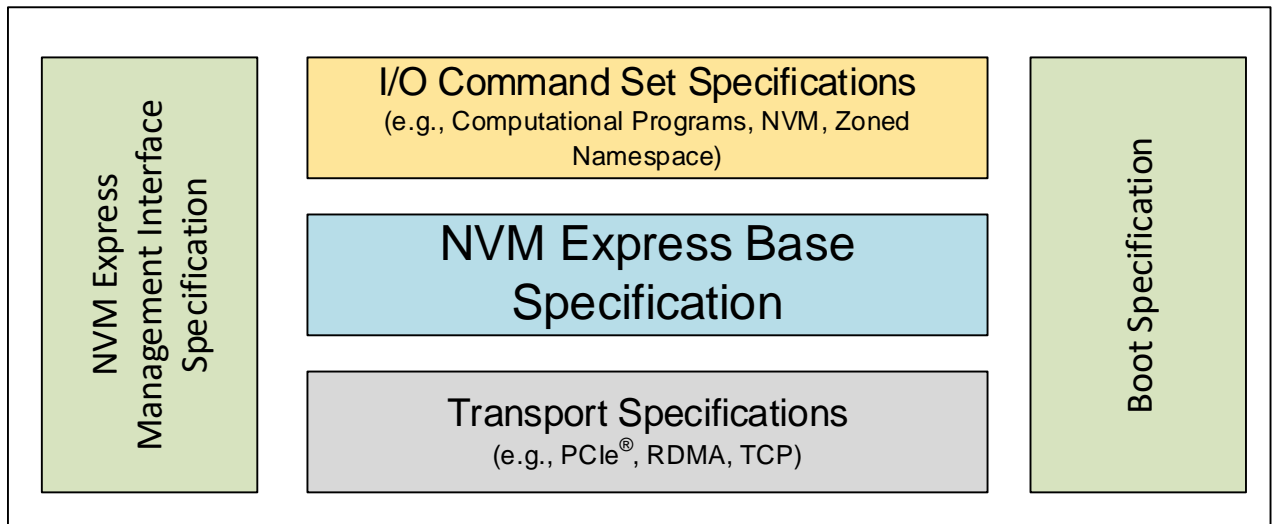


This specification supplements the NVM Express Base Specification. This specification defines additional data structures, Features, log pages, commands, and status values. This specification also defines extensions to existing data structures, features, log pages, commands, and status values. This specification defines requirements and behaviors that are specific to the Computational Programs Command Set. Functionality that is applicable generally to NVMe or that is applicable across multiple I/O Command Sets is defined in the NVM Express Base Specification.

If a conflict arises among requirements defined in different specifications, then a lower-numbered specification in the following list shall take precedence over a higher-numbered specification:

1. Non-NVMe specifications
2. NVM Express Base Specification
3. NVMe transport specifications
4. NVMe I/O Command Set specifications
5. NVM Express Management Interface Specification

### 1.3    Conventions

This specification conforms to the Conventions section, Keywords section, and Byte, Word, and Dword Relationships section of the NVM Express Base Specification.

### 1.4    Definitions

#### 1.4.1    Definitions from the NVM Express Base specification

This specification uses the definitions in the NVM Express Base Specification.

#### 1.4.2    Definitions specific to the Computational Programs Command Set

This section defines terms that are specific to this specification.

##### 1.4.2.1    compute namespace

A namespace that is associated with the Computational Programs Command Set.

##### 1.4.2.2    Namespace

Within this I/O Command Set specification, the term namespace without any additional qualifiers refers to a compute namespace.

##### 1.4.2.3    NVM Namespace

A namespace that is associated with the NVM Command Set or the Zoned Namespace Command Set.

##### 1.4.2.4    memory namespace

A namespace that is associated with the Subsystem Local Memory Command Set.

##### 1.4.2.5    Subsystem Local Memory (SLM)

Memory defined by the Subsystem Local Memory Command Set.

### 1.5    References

NVM Express Base Specification, Revision 2.0. Available from https://www.nvmexpress.org.

NVM Express Management Interface Specification, Revision 1.2. Available from https://www.nvmexpress.org.

# 2 Computational Programs Command Set Model

The NVM Express Base Specification defines a register level interface for host software to communicate with an NVM subsystem. This specification defines additional functionality for the Computational Programs Command Set.

The Computational Programs Command Set is an I/O Command Set that is used to discover, configure, and execute programs on a compute namespace. Data for the program is either in a memory namespace or in the Execute Program command data buffer. Data generated by the compute namespace is either returned in the CQE or is stored in a memory namespace. Data for program execution is:

- addressed by Memory Ranges (refer to section 2.1.2.1) that point to memory namespaces (refer to the Subsystem Local Memory (SLM) Command Set); or
- in the Execute Program command data buffer.

Each I/O Command Set is assigned a specific Command Set Identifier (CSI) value by the NVM Express Base Specification. The Computational Programs Command Set is assigned a CSI value of CSI-04h.

## 2.1 Theory of operation

The Computational Programs Command Set provides the mechanism to offload execution of programs from host to an NVM subsystem. Computational programs are functional pieces that achieve a well-defined purpose. These computational programs may be device-defined or downloaded from the host.

This specification defines a common framework for discovering, configuring, and executing computational programs on a namespace. This framework uses a host-driven approach, where programs operate on data in one or more memory namespaces, and program input and output data are staged into and out of the memory namespace(s) using NVMe commands issued by the host.

This I/O Command Set provides the following functionality:

- I/O commands
  - Execute Program
- Admin commands
  - Load Program
  - Program Activation Management
  - Memory Range Set Management
- Identify data structures and log pages
  - I/O Command Set specific Identify Controller data structures
  - I/O Command Set specific Identify Namespace data structures
  - Log Pages

Figure 2 shows the main architectural elements for this I/O Command Set, in an example that has all the different types of namespaces in the same domain of an NVM subsystem. Figure 2 also shows the different types of namespaces: compute namespaces, memory namespaces and NVM namespaces.

**Figure 2 - Architectural Block Diagram**
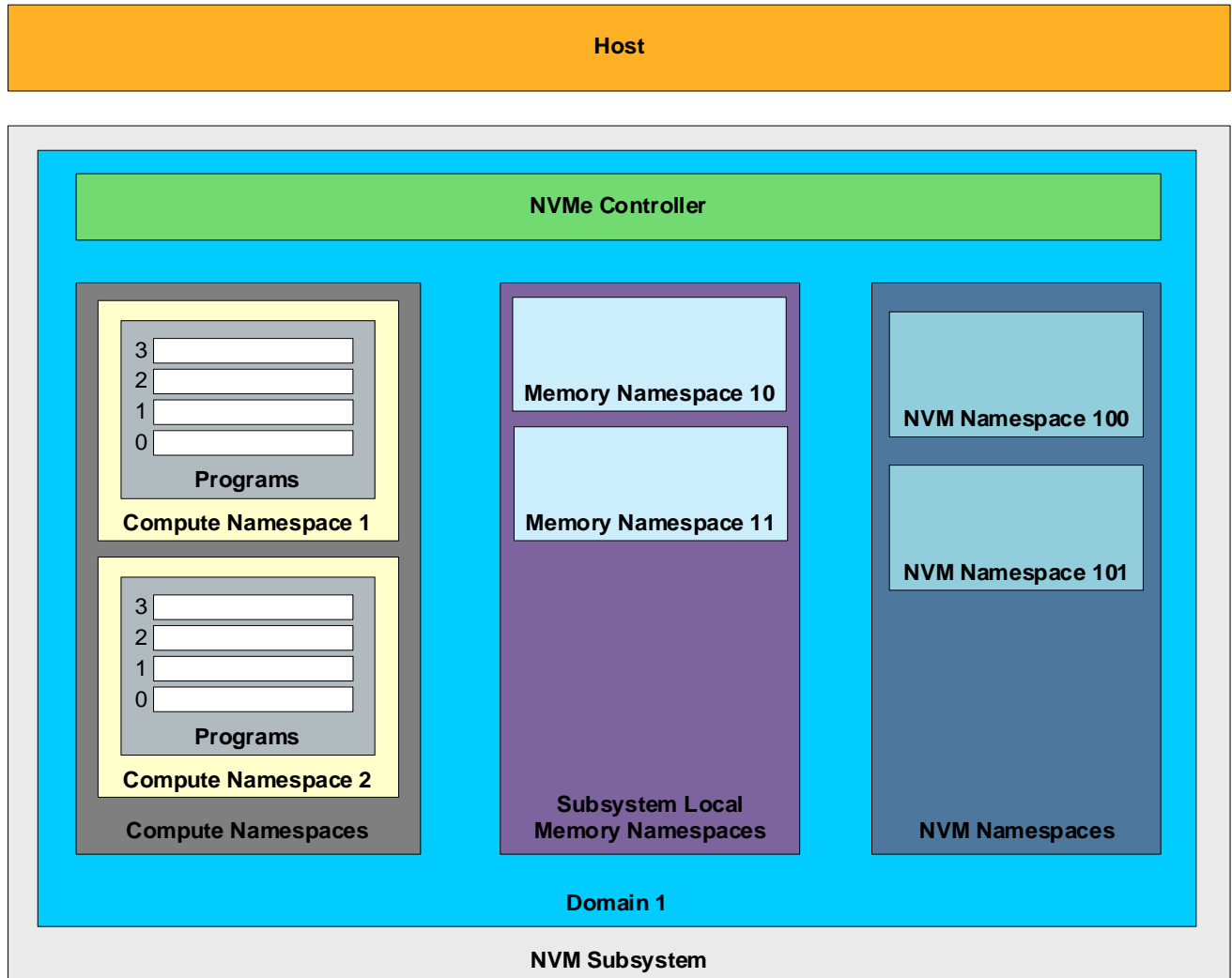


### 2.1.1 Storage Entities

Compute namespaces are not part of any Endurance Group or NVM Set (as defined in the Endurance Groups and NVM Sets sections of the NVM Express Base Specification).

### 2.1.2 Memory Model

Programs are able to use Subsystem Local Memory (SLM) for program input and output. SLM is divided into memory namespaces which are identified with an NSID, as defined in the SLM Command Set. SLM is addressed using a NSID, an offset, and a length. Refer to the SLM Command Set for more information about SLM. SLM is byte granular for accesses related to program execution and is dword granular for commands in the SLM Command Set.

To support computational programs, an NVM Subsystem contains zero or more memory namespaces that are reachable by compute namespaces. The ability of a compute namespace to reach a memory namespace is described by the Reachability Groups log page and the Reachability Associations log page as defined in the NVM Express Base Specification.

Note: Until the release of the next version of the NVM Express Base specification following 2.0, Reachability material is contained in TP4156.

Refer to section 2.1.3 for more information about compute namespaces.

The host uses:

- the SLM Command Set to manage the content of SLM that is used with computational programs; and
- the Memory Range Set Management command (refer to section 2.1.2.1 and section 4.2.2) and Memory Ranges specified in the data buffer of the Execute Program command (refer to section 3.2.1) to manage access to SLM.

### 2.1.2.1    Memory Range Sets

A Memory Range Set is one or more ranges of SLM. Each range of SLM is specified by a NSID, an offset, and a length. A Memory Range Set is created in a compute namespace and is specific to that compute namespace. Memory Range Sets are used for the purpose of limiting program access to a specific subset of SLM. A program is prohibited from accessing any SLM other than what is specified by:

- the Memory Range Set specified by the RSID field in the Execute Program command, if the RSID field is non-zero; or
- the Memory Ranges specified in the data buffer for the Execute Program command, if the RSID field is cleared to 0h.

Figure 3 shows an example of Memory Range Sets in two compute namespaces.

**Figure 3 - Memory Range Set Example**
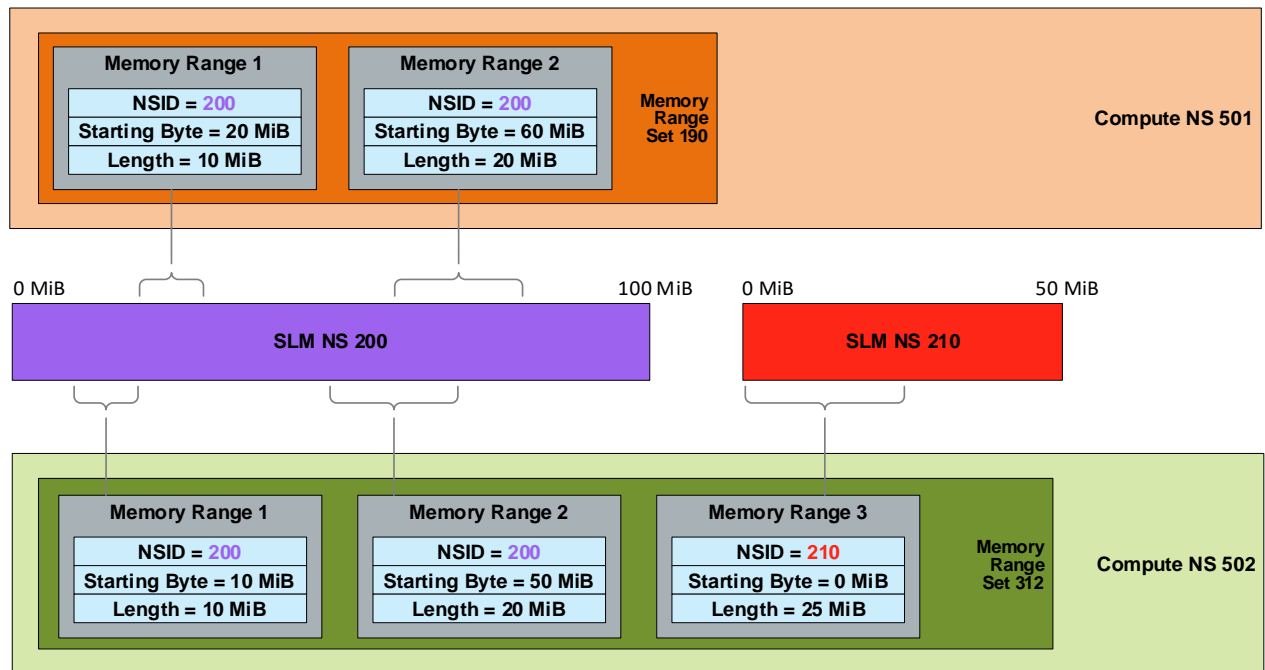
A memory namespace is only able to be included in a Memory Range Set if the NSID of the compute namespace in which the Memory Range Set is created and the NSID of that memory namespace are in the same Reachability Association. Figure 4 shows an example NVM Subsystem where:

- Any Memory Range Set defined in the compute namespace with NSID 20 is able to include memory namespaces with NSID of 10 and NSID of 11; and
- Any Memory Range Set defined in the compute namespace with NSID 22 or NSID 23 is able to include a memory namespace with NSID of 11.

**Figure 4: Reachability Architecture**
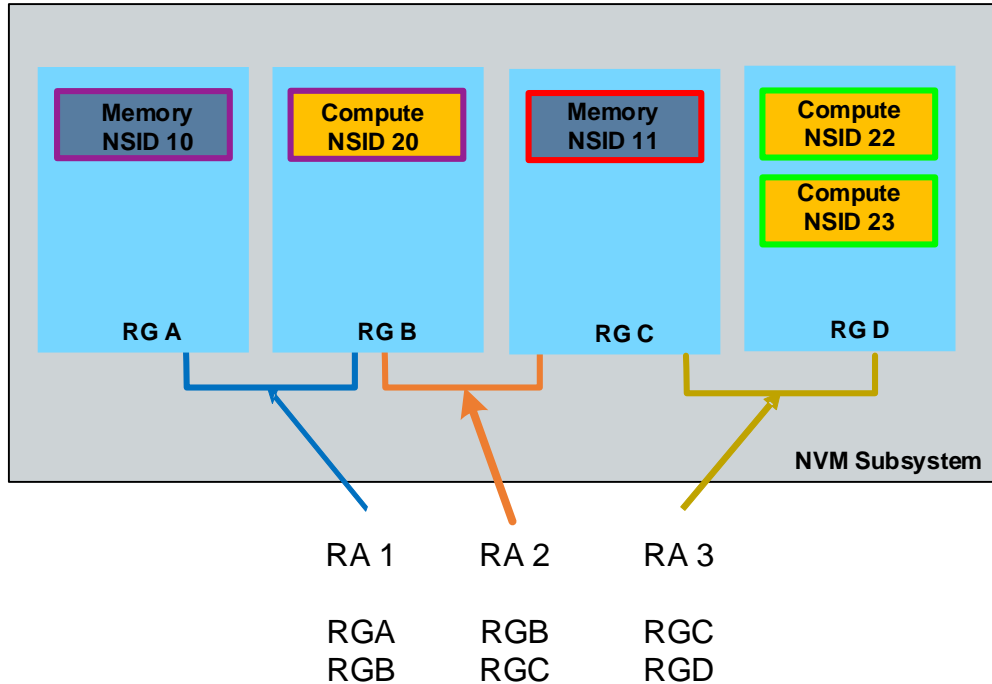


A Memory Range Set is associated with a particular program by an Execute Program command and a different Memory Range Set may be specified by other Execute Program commands that specify that same program.

Memory Range Sets are identified using a Memory Range Set ID as defined in section 4.2.2.1. The allocation of Memory Range Set IDs is managed on a per compute namespace basis. Memory Range Set ID values are required to be unique within a single compute namespace and are not required to be sequential within a single compute namespace. The value of 0h for a Memory Range Set ID is reserved.

Memory Range Sets are managed by the host using the Memory Range Set Management command. Different Memory Range Sets may refer to partially or fully overlapping ranges of data in a memory namespace to allow multiple programs to share access to the same data (refer to Figure 3). Memory Ranges within a Memory Range Set shall not overlap. A compute namespace may contain zero or more Memory Range Sets. Memory Range Sets from one compute namespace are unrelated to Memory Range Sets in any other compute namespace.

A compute namespace supports a particular granularity for Memory Range Set ranges. When creating a Memory Range Set, range granularities are required to conform to the Memory Range Granularity (refer to

section 4.2.2). Refer to the MRSG field in the I/O Command Set specific Identify Namespace data structure in Figure 36 for more details. Memory Ranges are specified using a Memory Range ID in the parameters for a program. The Memory Range ID of 0h refers to the data buffer for the Execute Program command and is not used by any Memory Range Set. Other Memory Range IDs are implied by the order of the Memory Ranges in the Memory Range Set with the first Memory Range in the Memory Range Set being associated with Memory Range ID 1h.

### 2.1.3   Compute Namespaces

A compute namespace is an entity in a domain that is able to execute one or more programs. A compute namespace is not associated with an Endurance Group or an NVM Set.

Each compute namespace may have different memory namespace reachability characteristics and is able to reach zero or more memory namespaces. Which memory namespaces are reachable by a namespace may be discovered using the Reachability Groups log page and the Reachability Associations log page (refer to section 2.1.3.1).

Each compute namespace supports one or more program types (refer to section 2.1.4.1).

Compute namespace characteristics may be discovered using the I/O Command Set specific Identify Namespace data structure (refer to Figure 36).

Programs are activated and deactivated on a compute namespace using the Program Activation Management command. Programs are executed on a compute namespace using the Execute Program command.

Compute namespaces may be discovered using the Identify command specifying CNS 02h to obtain the Active Namespace ID list.

### 2.1.3.1   Reachability

This I/O Command Set applies Reachability as defined in the NVM Express Base Specification to the operation of the Memory Range Set Management command and the Execute Program command.

Note: Until the release of the next version of the NVM Express Base specification following 2.0, Reachability material is contained in TP4156.

### 2.1.4   Program Management

### 2.1.4.1   Program Characteristics

There are multiple characteristics associated with a program. These characteristics include:

- whether a program is downloaded (refer to section 2.1.4.1.2) or device-defined (refer to section 2.1.4.1.3);
- the program type (e.g., eBPF, vendor specific. Refer to section 2.1.4.1.5); and
- what the program does.

A program may have an associated Program Unique Identifier (PUID) as defined by section 2.1.4.1.1.

Program characteristics are associated with a Program Index (PIND) on a specific compute namespace.

This I/O Command Set provides commands for loading, unloading, activating, deactivating, and executing downloadable programs and provides commands for activating, deactivating and executing device-defined programs. Programs operate on data in SLM, are activated with the Program Activation Management command, and executed with the Execute Program command.

A compute namespace may support downloaded programs with specific program types. A compute namespace may have device-defined programs. Information about the supported program characteristics for a namespace is obtained using:

- the Identify data structures; the Downloadable Program Types List log page (refer to section 4.1.4.2); and
- the Program List log page (refer to section 4.1.4.1).

### 2.1.4.1.1   Program Unique Identifier (PUID)

A PUID is a 64 bit value that may be used to uniquely identify the functionality provided by a program (e.g., a particular transcoding algorithm).

A PUID consists of an IEEE assigned organization identifier and a Universal Program Identifier. The IEEE assigned organization identifier is either 24 bits or 36 bits. The unique program functionality specified by the PUID is defined by the organization identified in the IEEE assigned organization identifier.

For NVMe standardized PUIDs, the IEEE organization identifier is set to DAE6D6h. For documentation of NVMe standardized PUIDs refer to the <NVMe Program Unique Identifier Registry> <this will be created as part of PUID TP4166>.

For organizations using a 36-bit IEEE organization identifier, the fields in the PUID are as shown in Figure 5.

**Figure 5: PUID Definition – 36-bit IEEE Organization Identifier**

| Bits | Definition |
|---|---|
| 35:0 | **Organization Identifier (OID):** IEEE assigned 36-bit organization identifier. |
| 63:36 | **Universal Program Identifier (UPI):** Identifier defined by the organization represented in the OID field. |

For organizations using a 24-bit IEEE organization identifier, the fields in the PUID are as shown in Figure 6.

**Figure 6: PUID Definition – 24-bit IEEE Organization Identifier**

| Bits | Definition |
|---|---|
| 23:0 | **Organization Identifier (OID):** IEEE assigned 24-bit organization identifier.<br><br>If the PUID is for a NVMe standardized program, then this field shall be set to DAE6D6h. |
| 63:24 | **Universal Program Identifier (UPI):** Identifier defined by the organization represented in the OID field.<br><br>If the PUID is for a NVMe standardized program, then this field shall be set to the identifier defined in the <NVMe Program Unique Identifier Registry> <this will be created as part of PUID TP4166> for that PUID. |

### 2.1.4.1.2   Downloadable Programs

Downloadable programs may be discovered, loaded, unloaded, activated, deactivated, and executed by the host. Defined downloadable program types are described in Figure 29.

A namespace may support zero or more downloadable program types. The host may determine the program types supported by the namespace as well as other related characteristics by issuing the Get Log Page command specifying the Downloadable Program Types List log page (refer to section 4.1.4.2) and examining the Program Type Descriptors.

The host downloads programs using the Load Program command on a namespace. The Load Program command may also be used to unload a previously downloaded program on a namespace. Downloaded programs are specific to a namespace. Downloaded programs may be validated as part of processing the Load Program command and/or as part of processing the Program Activation command. If the program is invalid, then that command is aborted with a status code of Invalid Program Data.

Downloading a program saves the program on the namespace but may not reserve compute resources for that program. Compute resources may be reserved by activating the program on that namespace (refer to section 2.1.4.2).

Programs may be constructed of multiple pieces that are individually downloaded with separate Load Program commands. Each Load Program command includes a Load Offset field (LOFF) and Number of Bytes field (NUMB) that specify a range of the program that the piece represents. The host should ensure that the LOFF field and NUMB field meet the alignment and granularity requirements indicated by the LPG field (refer to Figure 36). The first piece of a Program (i.e., LOFF field is cleared to 0) is required to be submitted prior to any other piece of the program. All other pieces of the program may be submitted in any order to the controller.

The Load Program command for the first piece of a program at a specific PIND uses the following fields (refer to section 4.2.1):

- Program Identifier Type (PIT);
- Program Type (PTYPE);
- Program Index (PIND);
- Program Identifier (PID);
- Program Size (PSIZE);
- Load Offset (LOFF), where the value of this field is cleared to 0h; and
- Number of Bytes (NUMB).

Subsequent Load Program commands for a program at a specific PIND use the following fields:

- PIND;
- LOFF, where the value of this field is non-zero; and
- NUMB.

The Load Program command for the first piece of a program at a specific PIND is required to have the LOFF field cleared to 0h. That Load Program command resets the contents of any buffers allocated to previous Load Program commands for that PIND. For programs that consist of multiple pieces, the host is required to wait for the first Load Program command to complete before submitting additional Load Program commands for that PIND. The PSIZE field is a byte granular value that specifies the total size of the program.

For a Load Program command with the LOFF field cleared to 0h, if the program type specified by the PTYPE field is not supported, then that command is aborted with a status code of Invalid Program Type.

The sum of the LOFF field and the NUMB field is required to be less than or equal to the value of the PSIZE field rounded up to the value in the LPG field (refer to Figure 36) on any Load Program command.

Each namespace may have limits related to downloadable programs (refer to I/O Command Set specific Identify Namespace data structure in Figure 38)).

### 2.1.4.1.3 Device-defined Programs

Device-defined programs are programs that are provided by the namespace and are not changeable by the host. These programs have defined functionality, inputs, and outputs. A namespace supports zero or more device-defined programs.

Device-defined programs may be discovered, activated/deactivated, and executed by the host. Unloading a device-defined program using the Load Program command is not supported. The functionality provided by an individual program is described using the PUID (refer to section 2.1.4.1) and is discovered by issuing the Get Log Page command and specifying the Program List log page (refer to section 4.1.4.1).

A device-defined program may be:

- an NVMe standardized program – Program where individual program semantics are defined in the <NVMe Program Unique Identifier Registry> <this will be created as part of PUID TP4166>; or

- a vendor-specific program – Program where individual program semantics are defined by a vendor.

### 2.1.4.1.4   Program Index

Programs are managed on a per compute namespace basis. A Program Index (PIND) is specific to a given compute namespace. Each program is associated with a specific PIND, and that PIND is used in commands to specify the program to be activated/deactivated, loaded/unloaded, or executed. PINDs are a logical association with a program. The physical location of programs is outside the scope of this specification. Programs may be discovered issuing the Get Log Page command specifying the Program List log page (refer to section 4.1.4.1).

Device-defined programs use PINDs that are not available for the host to use for downloadable programs.

Downloadable programs are associated with the PIND specified by the host in the Load Program command.

### 2.1.4.1.5   Program Type

The host may determine the program types supported by the namespace as well as other related characteristics by issuing the Get Log Page command and specifying the Downloadable Program Types List log page (refer to section 4.1.4.2) and examining the Program Type Descriptors.

### 2.1.4.2   Activating and Deactivating Programs

Program activation makes a program ready for a host to execute on a specific namespace, including reserving necessary compute resources, if applicable. Depending on the program type, activation may involve transpiling an eBPF program, flashing an FPGA with an RTL program, or other necessary actions. Doing these actions as a separate operation from program execution allows for more predictable latency for program execution, because these actions have been completed beforehand. There may be more programs on a namespace than are able to be activated at the same time. This allows more device-defined programs and downloaded programs on the namespace than are able to be activated and allows the host to choose which program(s) to activate.

Before a program is able to be executed on a compute namespace, the program is required to be activated on that compute namespace. The host activates or deactivates programs on a compute namespace using the Program Activation Management command.

The I/O Command Set specific Identify Namespace data structure (CNS 05h) (refer to Figure 36) may be used to discover limits to the number of activated programs on a specific namespace. A Get Log Page command specifying the Program List log page (refer to section 4.1.4.1) may be used to discover which programs have been activated.

### 2.1.5   Program Operation and Execution

Programs are executed on a namespace using the Execute Program command. The framework provided by the Execute Program command is described in this section.

Data that is specified by the host in the Execute Program command is passed as an input to the program. The internal format of parameter data is dependent on the program being executed. Parameter data size limits are defined by the I/O Command Set specific Identify Namespace data structure (refer to Figure 36).

A Memory Range Set is used for the purpose of limiting the executing program's access to a specific subset of SLM. One Memory Range Set may be specified in the Execute Program command. A Memory Range Set is only associated with a particular program for the duration of that Execute Program command, and a different Memory Range Set may be specified for other executions of the same program.

The Execute Program command returns a command status in the completion queue entry of the command. This status indicates whether the program was executed successfully by the namespace. In addition, a return value may be returned from the program itself, indicating program-specific result. Refer to section 3.2.1.1 for more details.

The program may output data to addresses in SLM. Once the Execute Program command completes, the host is able to:
- retrieve this data using commands to the appropriate memory namespace (e.g., a Read command to the memory namespace);
- use the data as input for a subsequent program; or
- use commands to copy the data to an NVM namespace (e.g., a Copy command to the NVM namespace).

Similar to commands submitted to NVM namespaces, the order of execution of queued Execute Program commands is not specified. If there are dependencies between multiple program executions, the host is responsible for coordination of the execution of the programs (e.g., waiting for completion of one execution before submitting another Execute Program command).

### 2.1.6 Example Flows

#### 2.1.6.1 Discovery of Compute Namespace Capabilities

1. The host reads the Controller Capabilities property and examines the CAP.CSS field to determine if multiple I/O command sets are supported.
2. The host issues an Identify command specifying CNS 1Ch to access the Identify I/O Command Set data structure (refer to the NVM Express Base Specification) to determine if this I/O Command Set is supported by the controller.
3. Methods to obtain a list of compute namespaces that are attached to the controller:

    a. The host issues an Identify command specifying CNS 2h to access the Active Namespace ID list (refer to the NVM Express Base Specification) to obtain the list of NSIDs for active namespaces on the controller. For each NSID, the host issues an Identify command specifying the Namespace Identification Descriptor List (i.e., CNS 3h). If the namespace is a compute namespace, then this returns the Computational Programs Command Set Identifier (i.e., 04h) in the NIDT.CSI field.; or

    b. The host may issue an Identify command specifying CNS 7h to access the I/O Command Set specific Active Namespace ID list (refer to the NVM Express Base Specification) specifying a CSI value of 04h to obtain a list of NSIDs associated with compute namespaces.

4. To obtain information about characteristics of compute namespaces, for each NSID that corresponds to a compute namespace, the host issues an Identify command specifying the I/O Command Set specific Identify Namespace data structure (refer to section 4.1.5.1).
5. To obtain information about programs for the compute namespace, the host issues a Get Log Page command specifying the Program List log page (refer to section 4.1.4.1).
6. To obtain information about downloadable program types supported by a compute namespace, the host issues a Get Log Page command specifying the Downloadable Program Types List log page (refer to section 4.1.4.2).
7. To obtain information about memory namespaces that are reachable by the compute namespace, the host issues a Get Log Page command specifying the Reachability Groups log page (refer to the NVM Express Base Specification) and issues a Get Log Page command specifying the Reachability Associations log page (refer to the NVM Express Base Specification).

Note: Until the release of the next version of the NVM Express Base specification following 2.0, Reachability material is contained in TP4156.

### 2.1.6.2    Load a Downloadable Program

**Figure 7: Load Program Flow**
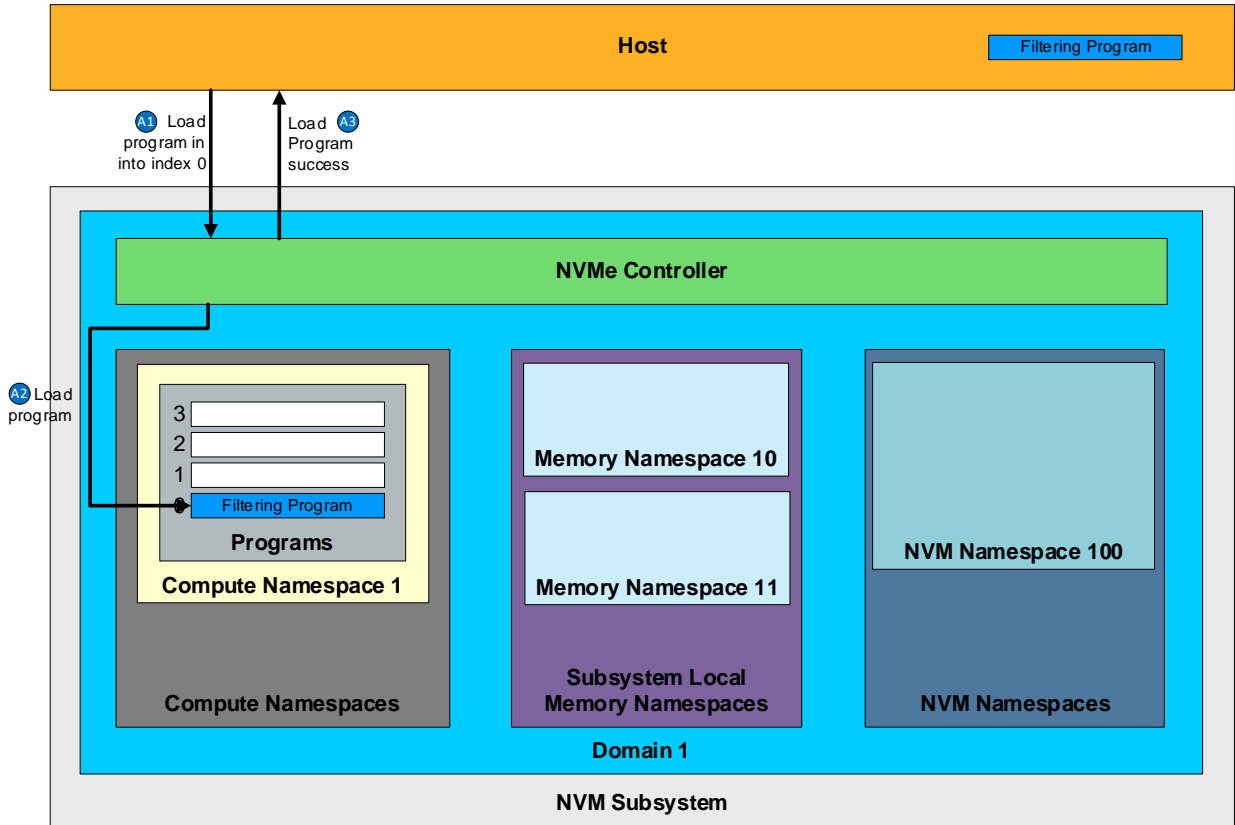


Prior to loading a program, the host determines the downloadable program types supported by the namespace and an available PIND for the program. Refer to section 2.1.6.1.

The process to download a program is:

1. The host issues a Load Program command to a compute namespace, specifying a PIND and a pointer to the executable program.

### 2.1.6.3   Activate a Program

**Figure 8: Program Activation Flow**
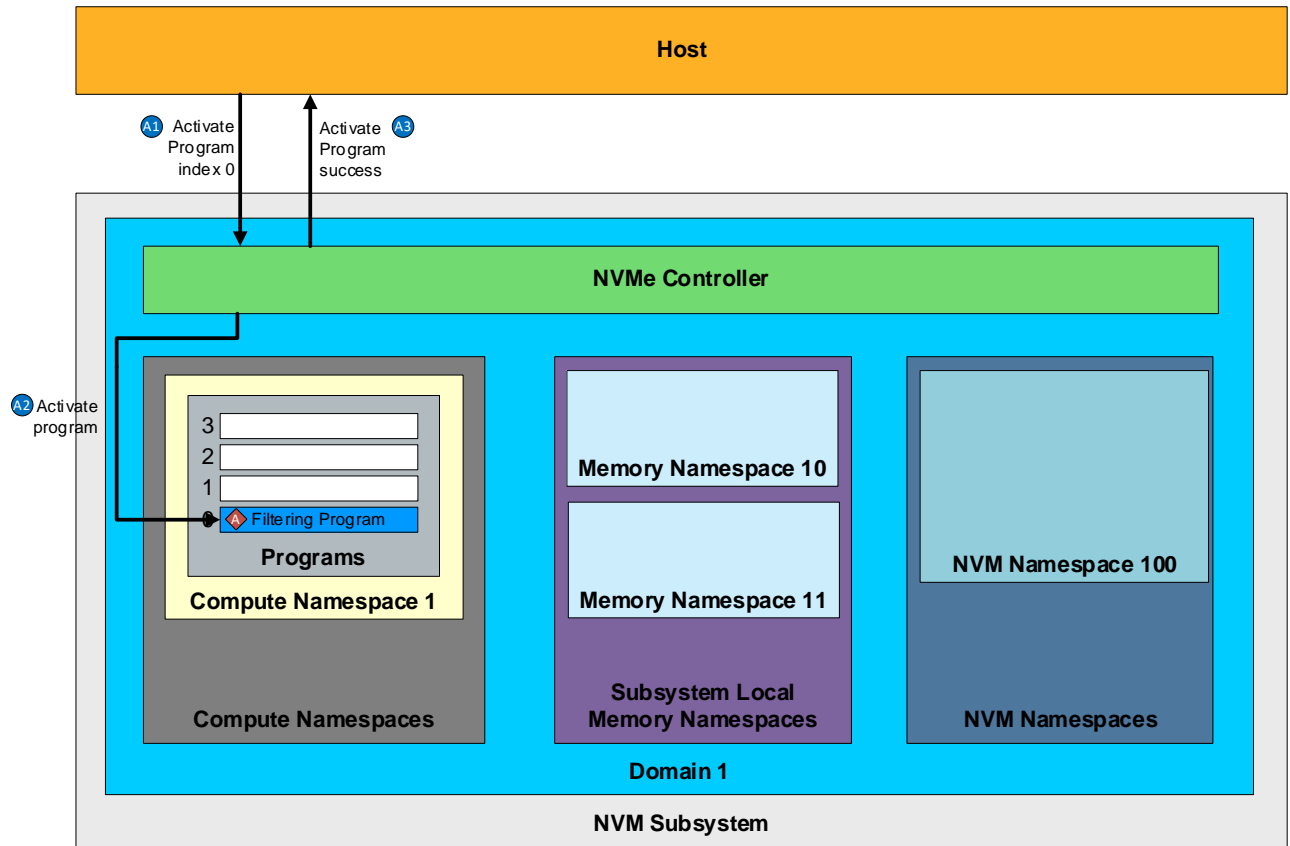


The process to activate a program is:

1. The host issues a Program Activation Management command to a compute namespace, specifying the PIND.

### 2.1.6.4    Create a Memory Range Set

**Figure 9: Create a Memory Range Set Flow**
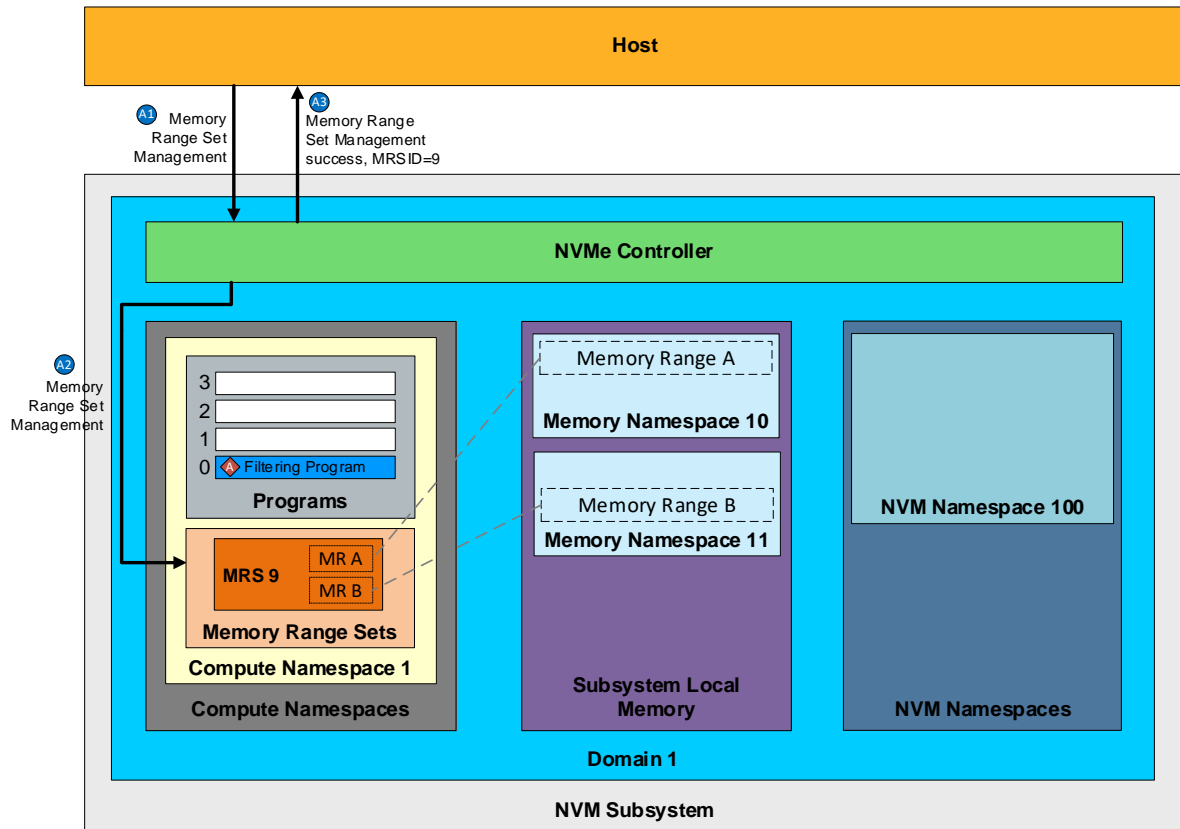


The process to create a Memory Range Set is:

1.  The host issues a Memory Range Set Management command to a compute namespace, specifying the create operation and the Memory Ranges.

    a.  Memory Ranges may only belong to memory namespaces that are reachable by the compute namespace. Refer to the Reachability Groups log page and the Reachability Associations log page in the NVM Express Base Specification.

2.  When the command completes, the host examines the posted completion queue entry for that command that contains the execution status and the Memory Range Set Identifier for the new Memory Range Set.

Note: Until the release of the next version of the NVM Express Base specification following 2.0, Reachability material is contained in TP4156.

### 2.1.6.5 Execute a Program

**Figure 10: Execute Program Flow**
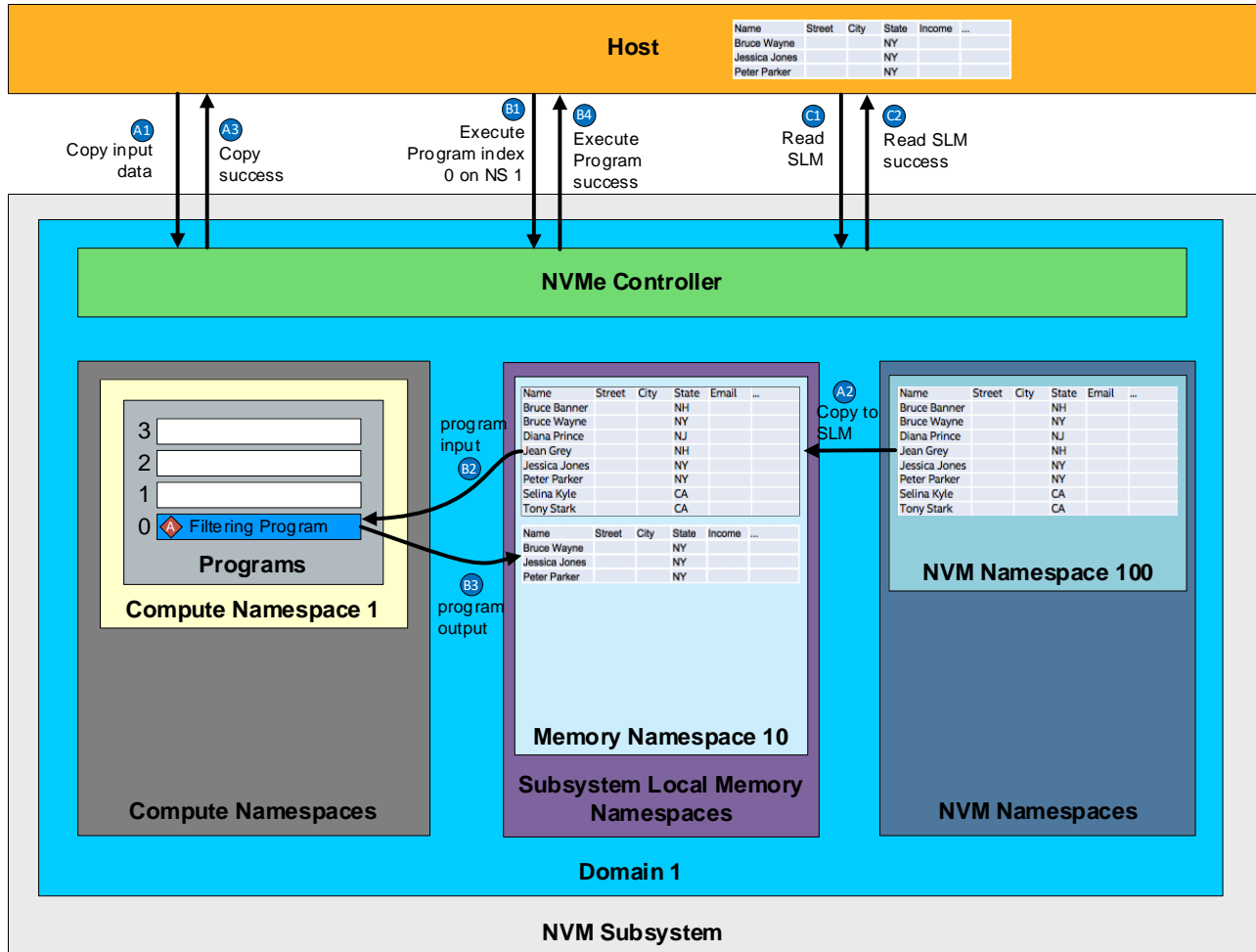


Prior to executing a program, the host is required to activate the program on the applicable namespace and have created an appropriate Memory Range Set.

The process to execute a program is:

1. The host issues one or more commands from the appropriate I/O command set to store program input data in the location in a memory namespace that is part of the Memory Range Set. In some cases, the input data may already have been placed into a memory namespace and this step may be skipped.
2. The host issues an Execute Program command to the compute namespace, specifying the following:
   a. PIND;
   b. Memory Range Set ID; and
   c. Parameter data.

3. The program is executed.

4. When the program completes, the host examines the posted completion queue entry for that command that contains the command execution status and return value from the program, if applicable.
5. In some cases, the host may transfer result data from a memory namespace to host memory by issuing a command to the appropriate memory namespace.

## 2.2 Flush Command

The Flush command (refer to the NVM Express Base Specification) has no effect on compute namespaces.

## 2.3 Command Ordering Requirements

This I/O Command Set does not impose any command ordering requirements beyond what is defined in the NVM Express Base Specification.

## 2.4 Fused Operation

This specification imposes no fused operation rules beyond what is defined in the NVM Express Base Specification.

## 2.5 Atomic Operation

This specification imposes no atomic operation rules beyond what is defined in the NVM Express Base Specification.

Accesses to SLM namespaces by programs invoked with the Execute Program command are atomic at a byte level, e.g., for a controller that experiences a failure while a program is writing multiple bytes, the contents of each byte are either all new data or all old data.

## 2.6 Metadata Region (MR)

This specification imposes no metadata region definition or rules beyond what is defined in the NVM Express Base Specification.

## 2.7 I/O Controller Requirements

## 2.7.1 Command Support

This I/O Command Set implements the command support requirements for I/O Controllers defined in the NVM Express Base Specification. Additionally, Figure 11 and Figure 12 defines which commands are mandatory, optional, and prohibited for an I/O controller that supports this I/O Command Set.

**Figure 11: I/O Controller – Admin Command Support**

| Command | Command Support Requirements [1] |
|---|---|
| Load Program | O |
| Program Activation Management | M |
| Memory Range Set Management | M |
| Notes: | |
| 1.   O = Optional, M = Mandatory, P = Prohibited | |

**Figure 12: I/O Controller – Computational Programs Command Set Support**

| Command | Command Support Requirements [1] |
|---|---|
| Execute Program | M |
| Notes: | |
| 1.    O = Optional, M = Mandatory, P = Prohibited | |

### 2.7.2    Log Page Support

This specification implements the log page support requirements for I/O Controllers defined in the NVM Express Base Specification. Additionally, Figure 13 defines Computational Programs Command Set specific definitions for log pages that are mandatory, optional, and prohibited for an I/O controller that supports this I/O Command Set.

**Figure 13: I/O Controller – Computational Programs Log Page Support**

| Log Page Name | Log Page Support Requirements [1] |
|---|---|
| Program List | M |
| Downloadable Program Types List | M |
| Memory Range Set List | M |
| Reachability Groups[2] | M |
| Reachability Associations[2] | M |
| Notes: | |
| 1.    O = Optional, M = Mandatory, P = Prohibited, NR = Not Recommended | |
| 2.    Defined in the NVM Express Base Specification | |

### 2.7.3    Features Support

This specification implements the feature support requirements for I/O Controllers defined in the NVM Express Base Specification.

# 3   I/O Commands for the Computational Programs Command Set

This section specifies I/O commands for this I/O Command Set.

## 3.1   Submission Queue Entry and Completion Queue Entry

The Submission Queue Entry (SQE) data structure and the fields that are common to all I/O Command Sets are defined in the Submission Queue Entry – Command Format section in the NVM Express Base Specification. The Completion Queue Entry (CQE) data structure and the fields that are common to all I/O Command Sets are defined in the Completion Queue Entry section in the NVM Express Base Specification. The command specific fields in the SQE and CQE structures (i.e., SQE Command Dword2, Dword 3, Dwords 10-15 and CQE Dword 0, and Dword 1) for this I/O Command Set are defined in this section.

### 3.1.1   Common Command Format

All commands in this command set follow the Common Command Format, as defined in the NVM Express Base Specification.

### 3.1.2   Computational Programs Specific Status Values

**Figure 14: Status Code – Command Specific Status Values, Computational Programs Command Set**

| Value | Description | Commands Affected |
|---|---|---|
| 8Ah | Insufficient Program Resources | Load Program |
| 8Bh | Invalid Memory Namespace | Memory Range Set Management |
| 8Ch | Invalid Memory Range Set | Memory Range Set Management |
| 8Dh | Invalid Memory Range Set Identifier | Memory Range Set Management, Execute Program |
| 8Eh | Invalid Program Data | Load Program, Program Activation Management |
| 8Fh | Invalid Program Index | Load Program, Program Activation Management, Execute Program |
| 90h | Invalid Program Type | Load Program |
| 91h | Maximum Memory Ranges Exceeded | Memory Range Set Management |
| 92h | Maximum Memory Range Sets Exceeded | Memory Range Set Management |
| 93h | Maximum Programs Activated | Program Activation Management |
| 94h | Maximum Program Bytes Exceeded | Load Program |
| 95h | Memory Range Set In Use | Memory Range Set Management |
| 96h | No Program | Load Program, Program Activation Management |
| 97h | Overlapping Memory Ranges | Memory Range Set Management |
| 98h | Program Not Activated | Execute Program |
| 99h | Program In Use | Load Program, Program Activation Management |
| 9Ah | Program Index Not Downloadable | Load Program |
| 9Bh | Program Too Big | Load Program |

## 3.2   Computational Programs Command Set Commands

This I/O Command Set includes the commands listed in Figure 15. Section 3.2 describes the definition of each of the commands defined by this specification.

**Figure 15: Opcodes for Computational Programs Command Set Commands**

| Opcode by Field | | | Combined Opcode[1] | Command[2] | Reference |
|---|---|---|---|---|---|
| (07) | (06:02) | (01:00) | | | |
| Standard Command | Function | Data Transfer[3] | | | |
| 0b | 000 00b | 01b | 01h | Execute Program | 3.2.1 |
| Notes: | | | | | |
| 1. Opcodes not listed are defined in the NVM Express Base Specification. | | | | | |
| 2. All Computational Programs Command Set commands use the NSID field. The value FFFFFFFFh is not supported in this field. | | | | | |
| 3. Indicates the data transfer direction of the command. All options to the command shall transfer data as specified or transfer no data. All commands, including vendor specific commands, shall follow this convention: 00b = no data transfer; 01b = host to controller; 10b = controller to host; 11b = bidirectional. | | | | | |

### 3.2.1 Execute Program command

This command executes a program on the specified namespace against the data in the specified Memory Range Set and with parameters provided in this command. Prior to execution, the program is required to be activated using the Program Activation Management command. The command uses Command Dword 2, Command Dword 3, Command Dword 4, Command Dword 10, Command Dword 11, Command Dword 12, and Command Dword 13 fields. All other Command Dwords are reserved.

Program parameter data may be passed in the Submission Queue Entry using the CPARAM1 field or CPARAM2 field, may be transferred from the host using the data buffer for this command, or both. Data described by DPTR is transferred to the namespace for use by the program. The format of program parameter data is program-specific.

If the RSID field is non-zero, then the program is only permitted to access the Memory Ranges described by the Memory Range Set specified in the RSID field. If the RSID field is cleared to 0h and the NUMR field is non-zero, then the Memory Ranges that the program is permitted to access are described in the data buffer for this command. If the RSID field is cleared to 0h and the NUMR field is cleared to 0h, then program data is in the data buffer for this command.

**Figure 16: Execute Program – Command Dword 2**

| Bits | Description |
|---|---|
| 31:16 | **Memory Range Set ID (RSID):** The identifier of the Memory Range Set the program is permitted to access. If this field is cleared to 0h and the NUMR field is non-zero, then the Memory Ranges that the program is permitted to access are specified in the data buffer. If this field is cleared to 0h and the NUMR field is cleared to zero, then the program is not permitted to access any Memory Ranges. |
| 15:00 | **Program Index (PIND):** This field specifies the index of the program to be executed. |

**Figure 17: Execute Program – Command Dword 3**

| Bits | Description |
|---|---|
| 31:00 | **Number of Memory Ranges (NUMR):** If the RSID field is cleared to 0h, this field specifies the number of Memory Ranges in the data buffer for this command. If the RSID field is non-zero and this field is non-zero, then the controller shall abort the command with a status code of Invalid Field in Command**.** |

**Figure 18: Execute Program – Command Dword 4**

| Bits | Description |
|---|---|
| 31:00 | **DPTR Data Length (DLEN):** The length of the data described by the DPTR field, in bytes. If this field is less than 32*NUMR then the controller shall abort the command with a status code of Invalid Field in Command. |

**Figure 19: Execute Program – Data Pointer**

| Bits | Description |
|---|---|
| 127:00 | **Data Pointer (DPTR):** This field specifies the location of a data buffer where data as defined in Figure 22 is transferred from if the DLEN field is non-zero. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field. |

**Figure 20: Execute Program – Command Dword 10, Command Dword 11**

| Bits | Description |
|---|---|
| 63:00 | **Parameter Data (CPARAM1):** This field specifies optional parameter data that may be passed in the command entry. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63:32. |

**Figure 21: Execute Program – Command Dword 12, Command Dword 13**

| Bits | Description |
|---|---|
| 63:00 | **Parameter Data (CPARAM2):** This field specifies optional parameter data that may be passed in the command entry. Command Dword 12 contains bits 31:00; Command Dword 13 contains bits 63:32. |

If Memory Ranges are passed in parameter data (i.e., the RSID field is cleared to 0h and the NUMR field is non-zero), the Memory Range ID is specified by the position of the Memory Range in the data buffer (e.g., the first Memory Range is associated with Memory Range ID 1). A memory namespace should only be included in a the Execute Program command if the NSID of the Execute Program command and the NSID of that memory namespace are in the same Reachability Association.

**Figure 22: Description of data buffer for the Execute Program command**

| NUMR | RSID | Data Buffer Description | | | |
|---|---|---|---|---|---|
| 0h | Any value | Data for use by the program (e.g., Parameter data and/or input data) | | | |
| non-zero | 0h | | **Bytes** | **Field** | **Description** |
| | | | 03:00 | Memory Namespace ID | Memory Range ID 1 |
| | | | 07:04 | Length | |
| | | | 15:08 | Starting Byte | |
| | | | 31:16 | Reserved | |
| | | | | | |
| | | | n+3:n[1] | Memory Namespace ID | Memory Range ID NUMR |
| | | | n+7:n+4 | Length | |
| | | | n+15:n+8 | Starting Byte | |
| | | | n+31:n+16 | Reserved | |
| | | | | | |
| | | | DLEN-1:NUMR*32[2] | | Data for use by the program (e.g., Parameter data and/or input data) |

**Figure 22: Description of data buffer for the Execute Program command**

| NUMR | RSID | Data Buffer Description |
|------|------|------------------------|
| non-zero | non-zero | Invalid (refer to Figure 17) |
| Notes: | | |
| 1. n=(NUMR-1)*32 | | |
| 2. If DLEN-1 = NUMR*32, there is no data for use by the program in the data buffer. | | |

### 3.2.1.1 Command completion

When the command is completed, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command. If the command completes with success, then Dword 0 and Dword 1 of the completion queue entry contains a value returned by the program, as shown in Figure 23.

**Figure 23: Execute Program – Completion Queue Entry Dword 0, Dword 1**

| Bits | Description |
|------|-------------|
| 63:00 | **Return Value (RVAL):** If the command is successful, this field contains a 64-bit value returned by the program. CQE Dword 0 contains bits 31:00; CQE Dword 1 contains bits 63:32. If the program does not support returning a value, then this field shall be cleared to 0h. |

**Figure 24: Execute Program – Command Specific Status Values**

| Value | Description |
|-------|-------------|
| 8Bh | **Invalid Memory Namespace:** The command references a Memory Namespace ID that does not correspond to any memory namespace, or the memory namespace is not reachable by this namespace. |
| 8Fh | **Invalid Program Index:** Specified index is not a valid value. |
| 8Dh | **Invalid Memory Range Set Identifier:** The command references a Memory Range Set ID that does not correspond to any Memory Range Set on the namespace. |
| 91h | **Maximum Memory Ranges Exceeded:** The command specified more than the maximum number of Memory Ranges supported by the namespace. |
| 96h | **No Program:** There is no program associated with the specified index. |
| 97h | **Overlapping Memory Ranges:** The command specified overlapping Memory Ranges. |
| 98h | **Program Not Activated:** Specified program has not been activated on the specified compute engine. |

# 4 Admin Commands for the Computational Programs Command Set

## 4.1 Admin Command behavior for the Computational Programs Command Set

The Admin Commands are as defined in the NVM Express Base Specification. The behavior for Admin Commands defined in this I/O Command Set is described in this section.

### 4.1.1 Asynchronous Event Request command

The Asynchronous Event Request command operates as defined in the NVM Express Base Specification.

### 4.1.2 Format NVM command

The Format NVM command is not supported by the this I/O Command Set. If the Format NVM command is directed to a compute namespace, the command shall be aborted with the Invalid Namespace or Format status code. If the Format NVM command is directed to an NVM subsystem or a controller with a compute namespace attached, there is no effect on compute namespaces and the command may complete successfully.

### 4.1.3 Get Features & Set Features commands

This I/O Command Set has no command set specific features.

### 4.1.4 Get Log Page command

The Get Log Page command operates as defined in the NVM Express Base Specification. In addition to the requirements in the NVM Express Base Specification, mandatory, optional, and prohibited Log Identifiers are defined in Figure 25. If a Get Log Page command is processed that specifies a Log Identifier that is not supported, then the controller should abort the command with a status code of Invalid Log Page.

In addition to the log pages defined in the NVM Express Base Specification, this I/O Command Set defines the log pages defined in this section. Log page scope is as defined in the NVM Express Base Specification, except as modified by this specification.

The rules for NSID usage are specified in the NVM Express Base Specification.

**Figure 25: Get Log Page – Log Page Identifiers**

| Log Identifier | Scope | Log Page Name | Reference |
|---|---|---|---|
| 82h | Namespace | Program List | 4.1.4.1 |
| 83h | Namespace | Downloadable Program Types List | 4.1.4.2 |
| 84h | Namespace | Memory Range Set List | 4.1.4.3 |

#### 4.1.4.1 Program List (Log Identifier 82h)

This log page is used to describe programs associated with a namespace. The log consists of a header describing the log page and a list of descriptors containing information about each program. The format for a Program descriptor is defined in Figure 27. The format for the Program List Log is defined in Figure 26. Program descriptors shall be returned in ascending PIND order.

The total number of descriptors is returned in the header of the log page. The Log Page Offset may be used to retrieve specific descriptors. If the Index Offset Supported bit is set to '1' in the LID Support and Effects data structure for this log page, then an entry is defined as a Program Descriptor as defined in Figure 27 (e.g., specifying an index offset of 2 returns this log page starting at the offset of Program Descriptor 1).

Activation status information returned in this log page is valid at the point in time the command was processed, but may have changed since then. Hosts are encouraged to keep track of activation status.

**Figure 26: Program List Log Page**

| Bytes | Description |
|---|---|
| **Header** ||
| 03:00 | **Number of Descriptors (NUMD):** Indicates the number of Program descriptors contained in the Log page. |
| 63:04 | Reserved |
| **Program Descriptor List** ||
| 127:64 | **Program 0 (P0)**: Contains the first Program descriptor as defined in Figure 27. |
| 191:128 | **Program 1 (P1)**: Contains the second Program descriptor as defined in Figure 27. |
| … | … |
| (((NUMD+1)*64) – 1): ((NUMD)*64) | **Program NUMD-1 (PNUMD-1)**: Contains the last Program descriptor as defined in Figure 27. |

**Figure 27: Program Descriptor Data Structure**

| Bytes | Description | | |
|---|---|---|---|
| 00 | **Bits** | **Description** | |
| | 07:06 | Reserved | |
| | 05:03 | **Program Identifier Type (PIT):** This field specifies the type of Program Identifier in the PID field. If the PEOCC field is cleared to 00b, then this field shall be cleared to 000b. |
| | | Value | Definition |
| | | 000b | PID is not used |
| | | 001b | PID is a Program Unique Identifier (PUID) (refer to section 2.1.4.1) |
| | | All Others | Reserved |
| | 02 | **Activation:** Indicates whether the program has been activated. If the PEOCC field is cleared to 00b, then this field shall be cleared to 0b. |
| | | Value | Definition |
| | | 0b | Not activated |
| | | 1b | Activated |
| | 01:00 | **Program Index Occupied (PEOCC):** |
| | | Value | Definition |
| | | 00b | Index does not contain a program. A program may be downloaded to this index. |
| | | 01b | Index contains a downloadable program. A program may be downloaded to this index. |
| | | 10b | Index contains a device-defined program. Entries containing device-defined programs shall not be overwritten or removed. |
| | | 11b | Reserved |
| 01 | **Program Type (PTYPE):** The type of the program. Bit values are defined in Figure 29. If the PEOCC field is cleared to 00b, then this field shall be cleared to 0h. | | |
| 07:02 | Reserved | | |

**Figure 27: Program Descriptor Data Structure**

| Bytes | Description |
|---|---|
| 15:08 | **Program ID (PID):** This field specifies an identifier for the program based on the value in the PIT field. If the PEOCC field is cleared to 00b, then this field shall be cleared to 0h.<table><tr><th>PIT Value</th><th>PI Definition</th></tr><tr><td>000b</td><td>Reserved</td></tr><tr><td>001b</td><td>**Program Unique ID (PUID):** The unique identifier of this program (refer to section 2.1.4.1).</td></tr><tr><td>All Others</td><td>Reserved</td></tr></table> |
| 63:16 | Reserved |

### 4.1.4.2 Downloadable Program Types List (Log Identifier 83h)

This log page is used to describe the downloadable program types associated with a namespace. The log page consists of a header describing the log page and a list of descriptors containing information about individual program types. The format for the Downloadable Program Types List log page is defined in Figure 26.

The total number of descriptors is returned in the header of the log page. The Log Page Offset may be used to retrieve specific descriptors. If the Index Offset Supported bit is cleared to '0' in the LID Support and Effects data structure for this log page, then an entry is defined as a Program Type Descriptor as defined in Figure 30 (e.g., specifying an index offset of 2 returns this log page starting at the offset of Downloadable Program Type Descriptor 1).

**Figure 28: Downloadable Program Types Log Page**

| Bytes | Description |
|---|---|
| **Header** | |
| 03:00 | **Number of Descriptors (NUMD):** Indicates the number of Program Type descriptors contained in the log. |
| 31:04 | Reserved |
| **Downloadable Program Type Descriptor List** | |
| 63:32 | **Downloadable Program Type Descriptor 0 (PTD0):** This field specifies the characteristics of the first program type. The format of this field is dependent on the PTYPE field in byte 0 of the descriptor. |
| 95:64 | **Downloadable Program Type Descriptor 1 (PTD1):** This field specifies the characteristics of the second program type. The format of this field is dependent on the PTYPE field in byte 0 of the descriptor. |
| ... | ... |
| (((NUMD+1)*32) – 1): ((NUMD)*32) | **Downloadable Program Type Descriptor NUMD-1 (PTDNUMD-1):** This field specifies the characteristics of the last program type. The format of this field is dependent on the PTYPE field in byte 0 of the descriptor. |

Each program type has a unique PTYPE value. Refer to Figure 29 for PTYPE values.

**Figure 29: Program Type Values**

| Value | Program Type Description | Identify Program Type Descriptor Data Structure |
|---|---|---|
| 0h | Device-defined | Not applicable. Only downloadable program types have a program type descriptor data structure. |
| 01h to 7Fh | Reserved | None |
| C0h to FFh | Vendor-specific downloadable programs | Refer to Figure 30. |

Figure 30 defines the format of the Downloadable Program Type Descriptor data structure.

**Figure 30: Downloadable Program Type Descriptor Data Structure**

| Bytes | Description |
|---|---|
| 00 | **Program Type (PTYPE):** Program type, as defined in Figure 29.<br><br>This field is mandatory in all Downloadable Program Type Descriptor data structures. The value 0h is reserved. |
| 01 | **Version (VER):** This field contains the version of this data structure and shall be cleared to 0h. |
| 07:02 | Reserved |
| 31:08 | **Specific Program Type:** The definition of this field is specific to each program type. Refer to the definition of the specified program type in the PTYPE field for the definition of this field for that program type. |

### 4.1.4.3   Memory Range Set List (Log Identifier 84h)

This log page is used to describe the Memory Range Sets associated with a namespace. The log page consists of a header describing the log page and a list of descriptors containing information about individual Memory Range Sets. The format for an individual Memory Range Set descriptor is defined in Figure 33. The format for the Memory Range Set List log page is defined in Figure 32. Memory Range Set descriptors shall be returned in ascending Memory Range Set ID (RSID) order. The Memory Range Set List log page shall always contain a descriptor for the Memory Range Set ID 0 (refer to section 2.1.2.1).

The total number of descriptors is returned in the header of the log page. The Log Page Offset may be used to retrieve specific descriptors. If the Index Offset Supported bit is cleared to '0' in the LID Support and Effects data structure for this log page, then an entry is defined as a Memory Range Set Descriptor as defined in Figure 33 (e.g., specifying an index offset of 2 returns this log page starting at the offset of Memory Range Set 1).

The Memory Range Set List returned in this log page is valid at the point in time the command was processed, but may have changed since then. Hosts are encouraged to keep track of Memory Range Sets.

The Log Specific Parameter field in Command Dword 10 (refer to NVM Express Base Specification) for this log page is defined in Figure 31.

**Figure 31: Memory Range Set List Log Specific Parameter Field**

| Bits | Description |
|---|---|
| 14:09 | Reserved |
| 08 | **Return Identifiers Only (RIO):** If this bit is set to '1', then the controller shall return Memory Range Set Descriptors with the Number of Memory Range Descriptors field in each Memory Range Set Descriptor cleared to 0h (i.e., no Memory Range Descriptors are returned). If this bit is cleared to '0', then the controller shall return Memory Range Descriptors that contain Memory Ranges that make up the Memory Range Set described by that Memory Range Set Descriptor and the Number of Memory Range Descriptors field set to the number of Memory Ranges in that Memory Range Set. |

**Figure 32: Memory Range Set List Log Page**

| Bytes | Description |
|---|---|
| | **Header** |
| 03:00 | **Number of Descriptors (NUMD):** Indicates the number of Memory Range Set descriptors contained in the log. This field shall not be cleared to 0h. |

**Figure 32: Memory Range Set List Log Page**

| Bytes | Description |
|---|---|
| | **Memory Range Set Descriptor List** |
| n:04 | **Memory Range Set 0 (MRS0):** Contains the first Memory Range Set descriptor as defined in Figure 33. |
| m:n+1 | **Memory Range Set 1 (MRS1):** Contains the second Memory Range Set descriptor as defined in Figure 33. |
| … | … |
| x:y | **Memory Range Set NUMD-1 (MRSNUMD-1):** Contains the last Memory Range Set descriptor as defined in Figure 33. |

**Figure 33: Memory Range Set Descriptor Data Structure**

| Bytes | Description |
|---|---|
| 01:00 | **Memory Range Set ID (RSID):** The Memory Range Set identifier. |
| 05:02 | **Number of Memory Range Descriptors (NMR):** This field specifies the number of Memory Range descriptors in this Memory Range Set Descriptor. If the RIO bit is set to '1', then this field shall be cleared to 0h. |
| 31:16 | Reserved |
| 63:32 | **Memory Range Descriptor 1:** Contains the first Memory Range Descriptor for Memory Range ID 1 as defined in Figure 34, if any. |
| 95:64 | **Memory Range Descriptor 2:** Contains the second Memory Range Descriptor for Memory Range ID 2 as defined in Figure 34, if any. |
| … | … |
| $(((NMR+2)*32) - 1)$: $((NMR+1)*32)$ | **Memory Range Descriptor NMR:** Contains the last Memory Range Descriptor for Memory Range ID NMR as defined in Figure 34, if any. |

**Figure 34: Memory Range Descriptor Data Structure**

| Bytes | Description |
|---|---|
| 03:00 | **Memory Namespace ID (MNSID):** This field specifies the NSID for the memory namespace containing the specified range. |
| 07:04 | **Length:** This field specifies the length of the range in bytes. |
| 15:08 | **Starting Byte:** This field specifies the starting byte of the Memory Range. |
| 31:16 | Reserved |

## 4.1.5   Identify Command

This specification implements the Identify command and associated Identify data structures defined in the NVM Express Base Specification. Additionally, this I/O Command Set specifies the data structures defined in this section.

**Figure 35: Identify – CNS Values**

| CNS Value | O/M [1] | Definition | NSID [2] | CNTID [3] | CSI [4] | Reference Section |
|---|---|---|---|---|---|---|
| | | **Active Namespace Management** | | | | |
| 05h | M [5] | Identify I/O Command Set specific Namespace data structure for the specified NSID for the I/O Command Set specified in the CSI field. | Y | N | Y | 4.1.5.1 |

| CNS Value | O/M [1] | Definition | NSID [2] | CNTID [3] | CSI [4] | Reference Section |
|---|---|---|---|---|---|---|
| 06h | M | Identify I/O Command Set specific Controller data structure for the controller processing the command. | Y | N | Y | 4.1.5.2 |
| Notes: 1. O/M definition: O = Optional, M = Mandatory. 2. The NSID field is used: Y = Yes, N = No. 3. The CDW10.CNTID field is used: Y = Yes, N = No. 4. The CDW11.CSI field is used: Y = Yes, N = No. 5. Mandatory for controllers that support the Namespace Management capability (refer to the NVM Express Base Specification). | | | | | | |

#### 4.1.5.1 I/O Command Set specific Identify Namespace data structure (CNS 05h)

Figure 36 defines the I/O Command Set specific Identify Namespace data structure for this I/O Command Set. This data structure is used to provide information about a compute namespace.

For the list of memory namespaces that are reachable by this namespace, refer to the Reachability Groups log page and the Reachability Associations log page in the NVM Express Base Specification.

Note: Until the release of the next version of the NVM Express Base specification following 2.0, Reachability material is contained in TP4156.

**Figure 36: I/O Command Set specific Identify Namespace data structure for the Computational Programs Command Set**

| Bytes | Description | Capability Field |
|---|---|---|
| **Activated programs** | | |
| 01:00 | **Maximum Activated Programs (MAXACT):** Indicates the maximum number of activated programs for this compute namespace. If this field is cleared to 0h, the namespace does not impose a limit on the number of activated programs. | **No** |
| **Memory Range Sets** | | |
| 03:02 | **Maximum Number of Memory Range Sets (MAXMEMRS):** Indicates the maximum number of Memory Range Sets. If this field is cleared to 0h, the namespace does not impose a limit on the number of Memory Range Sets. | **No** |
| 05:04 | **Memory Range Size Granularity (MRSG):** This field specifies the granularity of the size of each Memory Range. The Memory Range size granularity is (2 ^ MRSG) bytes. This granularity shall not be smaller than the granularity of all memory namespaces reachable by this namespace (refer to the Namespace Optimal Write Granularity field in the Identify Namespace data structure for the SLM Command Set. | **No** |
| 06 | **Maximum Number of Ranges in a Memory Range Set (MAXMEMR):** Indicates the maximum number of ranges in a Memory Range Set. If this field is cleared to 0h, the namespace does not impose a limit on the number of ranges. | **No** |
| 07 | Reserved | |
| **Downloadable programs** | | |
| 15:08 | **Maximum Bytes for All Downloaded Programs (MAXPB):** Indicates the maximum bytes for all downloaded programs, in MiB. If this field is cleared to 0h, the namespace does not impose a limit. | **No** |
| 16 | **Load Program Granularity (LPG):** Indicates granularity required for all pieces of a downloaded program. The Load Program granularity is (2 ^ LPG) bytes. | **No** |
| 4095:17 | Reserved | |

#### 4.1.5.2 Identify I/O Command Set specific Controller data structure (CNS 06h, CSI 04h)

Figure 37 defines the I/O Command Set specific Identify Controller data structure for this I/O Command Set.

**Figure 37: Identify Controller Data Structure, Computational Programs Command Set Dependent – General**

| Bytes | O/M | Description |
|-------|-----|-------------|
| 03:00 | M | **Version (VER):** This field contains a Specification Version Descriptor (refer to the NVM Express Base Specification) indicating the version of this specification supported by the controller, as defined in Figure 38. |
| 4095:04 | | Reserved |

Published versions of this specification and the values that shall be reported by compliant controllers are defined in Figure 38.

**Figure 38: Computational Programs Command Set Specification Version Descriptor Field Values**

| Specification Versions [1] | MJR Field | MNR Field | TER Field |
|----------------------------|-----------|-----------|-----------|
| 1.0 | 1h | 0h | 0h |
| Notes: | | | |
| 1. The specification version listed includes lettered versions (e.g., 1.0 includes 1.0, 1.0a, 1.0b, etc.). | | | |

### 4.1.6    Namespace Attachment command

The Namespace Attachment command operates as defined in the NVM Express Base Specification.

### 4.1.7    Namespace Management command

The Namespace Management command is not supported for compute namespaces. If a Namespace Management command specifies this I/O Command Set (i.e., CSI 04h), then the controller shall abort that command with a status code of Invalid Field in Command.

### 4.1.8    Sanitize command

Sanitize deletes downloaded programs and removes associated PIND entries. It also removes any program-related remnants or parameters that may be in the internal compute environment of the namespace. Sanitize does not affect Memory Range Sets that have been created by the host. Sanitize does not remove program activation settings for device-defined programs. During a sanitize operation, all Admin commands and I/O commands defined by this I/O Command Set shall be aborted as defined in the NVM Express Base Specification. Block Erase, Crypto Erase, and Overwrite sanitize operation types have the same effect on compute namespaces.

## 4.2    I/O Command Set Specific Admin commands

The Admin commands are as defined in the NVM Express Base Specification. The Admin commands for this I/O Command Set are defined in this section.

### 4.2.1    Load Program command

The Load Program command is used to load a downloadable program and associate the program with the specified PIND, or unload the program associated with the specified PIND. This command uses Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15 fields. All other Command Dwords are reserved.

It is not necessary to unload the program associated with an index prior to loading a new program with that index, however the command may be aborted if an Execute Program command referencing that index is in progress.

The Load Program command shall only be used to load a program of a type that is supported by the namespace. The host may determine the program types supported by the namespace as well as other related characteristics by issuing the Get Log Page command specifying the Downloadable Program Types List log page (refer to section 4.1.4.2).

The host may determine programs associated with each PIND by examining returned data from issuing a Get Log Page command specifying the Program List log page (refer to section 4.1.4.1). Programs are managed on a per-namespace basis, meaning each namespace has its own set of programs.

Programs that have been loaded by the host are not persistent, and are cleared on power on resets.

**Figure 39: Load Program – Data Pointer**

| Bits | Description |
|------|-------------|
| 127:00 | **Data Pointer (DPTR):** This field specifies the location of a data buffer where program data is transferred from. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field. This field is reserved if the SEL bit is set to '1'. |

**Figure 40: Load Program – Command Dword 10**

| Bits | Description |
|------|-------------|
| 31:28 | Reserved |
| 27:25 | **Program Identifier Type (PIT):** This field specifies the interpretation of the PID value in this SQE.<table><tr><td>Value</td><td>Definition</td></tr><tr><td>000b</td><td>PID is not used</td></tr><tr><td>001b</td><td>PID is a Program Unique Identifier (PUID) (refer to section 2.1.4.1)</td></tr><tr><td>All Others</td><td>Reserved</td></tr></table> |
| 24 | **Select (SEL):** This field selects the type of operation to perform.<table><tr><td>Value</td><td>Description</td></tr><tr><td>0b</td><td>Load the program described by the DPTR field into the index specified by PIND</td></tr><tr><td>1b</td><td>Unload the program specified by PIND</td></tr></table> |
| 23:16 | **Program Type (PTYPE):** The type of program. Namespaces may support a subset of all program types. Program type bit values are defined in Figure 29.<br><br>This field is reserved if the SEL bit is set to '1'. |
| 15:00 | **Program Index (PIND):** This field specifies the index of the program that this command applies to.<br><br>If the SEL bit is set to '1', a value of FFFFh in this field shall unload all programs for the specified namespace. If the SEL field is cleared to '0', a value of FFFFh in this field shall return an error of Invalid Program Index. |

**Figure 41: Load Program – Command Dword 11**

| Bits | Description |
|------|-------------|
| 31:00 | **Program Size (PSIZE):** This field specifies the total size of the program in bytes. This field is reserved if the SEL bit is set to '1'. |

**Figure 42: Load Program – Command Dword 12, Command Dword 13**

| Bits | Description |
|------|-------------|
| 64:00 | **Program Identifier (PID):** This field specifies a Program Identifier based on the value of the PIT field in this SQE. This field is reserved if the SEL bit is set to '1'. <br><br> <table><tr><td>**PIT Value**</td><td>**Program Identifier Definition**</td></tr><tr><td>000b</td><td>Reserved</td></tr><tr><td>001b</td><td>**Program Unique ID (PUID):** The unique identifier of this program (refer to section 2.1.4.1).</td></tr><tr><td>All Others</td><td>Reserved</td></tr></table> |

**Figure 43: Load Program – Command Dword 14**

| Bits | Description |
|------|-------------|
| 31:00 | **Number of Bytes (NUMB):** This field specifies the number of bytes to transfer. This field shall be an integer multiple of dwords (i.e., bits 01:00 cleared to 00b). If bits 1:0 are non-zero, then the controller shall abort the command with a status code of Invalid Field in Command. This field is required to be an integer multiple of the value in the LPG. Field (refer to Figure 36). This field is reserved if the SEL bit is set to '1'. |

**Figure 44: Load Program – Command Dword 15**

| Bits | Description |
|------|-------------|
| 31:00 | **Load Offset (LOFF):** This field specifies the byte offset of the beginning of the data to transfer for this Load Program command. This field shall be an integer multiple of dwords (i.e., bits 01:00 cleared to 00b). If bits 1:0 are non-zero, then the controller shall abort the command with a status code of Invalid Field in Command.This fieldis required to be an integer multiple of the value of the LPG field (refer to Figure 36). This field is reserved if the SEL bit is set to '1'. |

#### 4.2.1.1   Command completion

Upon completion of the Load Program command, the controller posts a completion queue entry to the Admin Completion Queue.

**Figure 45: Load Program – Command Specific Status Values**

| Value | Description |
|-------|-------------|
| 8Ah | **Insufficient Program Resources:** Insufficient resources to load program. |
| 8Eh | **Invalid Program Data:** Program data is not valid. |
| 8Fh | **Invalid Program Index:** Specified index is not a valid value. |
| 90h | **Invalid Program Type:** Program type is not a valid value, or program type is not supported. |
| 94h | **Maximum Program Bytes Exceeded:** Loading the program may exceed the limit of maximum program bytes for the program type. |
| 96h | **No Program:** There is no program associated with the specified index. |
| 99h | **Program In Use:** Cannot perform the unload operation since a command referencing this program is in progress. |
| 9Ah | **Program Index Not Downloadable:** Program with the specified index cannot be loaded or unloaded. |
| 9Bh | **Program Too Big:** Program size exceeds the maximum bytes for the program type. |

### 4.2.2   Memory Range Set Management command

The Memory Range Set Management command is used by the host to manage Memory Range Sets, including create and delete operations. The command uses Command Dword 10 and Command Dword 11. All other Command Dwords are reserved.

Memory Range Sets are managed on a per-namespace basis.

Memory Range Sets are not persistent, and shall be cleared on power on resets.

The Memory Range Set ID (RSID) field is used as follows for individual operations:

- Create: The RSID field is reserved for this operation; host clears this field to a value of 0h. The namespace shall select an available Memory Range Set Identifier to use for the operation. A Memory Range Set ID of 0 shall not be returned by a controller in response to a create operation; or
- Delete: This field specifies the previously created Memory Range Set to delete in this operation. Specifying a value of FFFFh is used to delete all Memory Range Sets in the namespace. If the value of FFFFh is specified and there are zero Memory Range Sets, the command shall complete successfully.

**Figure 46: Memory Range Set Management – Data Pointer**

| Bits | Description |
|---|---|
| 127:00 | **Data Pointer (DPTR):** This field specifies the data buffer to use for this command. Refer to the Common Command Format definition in the NVM Express Base Specification for the definition of this field. |

**Figure 47: Memory Range Set Management – Command Dword 10**

| Bits | Description |
|---|---|
| 31:16 | **Memory Range Set ID (RSID):** The Memory Range Set identifier. |
| 15:04 | Reserved |
| 03:00 | **Select (SEL):** This field selects the type of management operation to perform. <br><br> | Value | Description | <br> | 0h | Create | <br> | 1h | Delete | <br> | 2h to Fh | Reserved | |

**Figure 48: Memory Range Set Management – Command Dword 11**

| Bits | Description |
|---|---|
| 31:08 | Reserved |
| 07:00 | **Number of Memory Ranges (NUMR):** For a create operation (i.e., the SEL field is cleared to 0h), this field specifies the number of Memory Range Descriptors that are specified in the command. If the value in this field is cleared to 0h or is greater than 128, then the controller shall abort the command with a status code of Invalid Field in Command. For all other operations this field is reserved. |

For a create operation, the data buffer specifies a list of Memory Ranges to be created. Each Memory Range consists of a Memory Namespace ID, a length in bytes, and a starting byte for that Memory Range as specified in Figure 34. The definition of the Memory Range Set Management command Memory Range Definitions is specified in Figure 49. The maximum number of Memory Ranges is 128.

Namespace limits on granularity of Memory Ranges may be found in the I/O Command Set specific Identify Namespace data structure (refer to Figure 38). All memory NSIDs specified in the data buffer for a create operation of the Memory Range Set Management command shall be memory namespaces that are reachable by the specified namespace of that Memory Range Set Management command. If Memory Ranges overlap, the controller shall abort the command with a status code of Overlapping Memory Ranges.

**Figure 49: Create operation – Memory Range Definitions**

| Range | Bytes | Field |
|---|---|---|
| Memory Range ID 1 | 31:00 | **Memory Range Descriptor 1:** Contains the Memory Range Descriptor for Memory Range ID 1 as defined in Figure 34, if any. |
| Memory Range ID 2 | 63:32 | **Memory Range Descriptor 2:** Contains the Memory Range Descriptor for Memory Range ID 2 as defined in Figure 34, if any. |
| … | | |
| Memory Range ID 128 | 4095:4064 | **Memory Range Descriptor 128:** Contains the Memory Range Descriptor for Memory Range ID 128 as defined in Figure 34, if any. |

#### 4.2.2.1 Command completion

Upon completion of the command, the controller posts a completion queue entry to the associated Completion Queue indicating the status for the command. For successful create operations, the 16-bit Memory Range Set ID of the new Memory Range Set is returned as specified in Figure 50. For delete operations, Dword 0 of the completion queue entry is reserved.

**Figure 50: Create operation – Completion Queue Entry Dword 0**

| Bits | Description |
|---|---|
| 31:16 | Reserved |
| 15:00 | **Memory Range Set Identifier:** For a create operation (i.e., the SEL field is cleared to 0h), this field indicates the Memory Range Set Identifier of the Memory Range Set created by the command. For all other operations this field is reserved. |

**Figure 51: Memory Range Set Management – Command Specific Status Values**

| Value | Description |
|---|---|
| 8Bh | **Invalid Memory Namespace:** The command references a Memory Namespace ID that does not correspond to any memory namespace, or the memory namespace is not reachable by this namespace. |
| 8Ch | **Invalid Memory Range Set:** The command specifies a Memory Range that does not conform to the Memory Range granularity. |
| 8Dh | **Invalid Memory Range Set Identifier:** The command references a Memory Range Set ID that does not correspond to any Memory Range Set on the namespace. |
| 91h | **Maximum Memory Ranges Exceeded:** The command specified more than the maximum number of Memory Ranges supported by the namespace. |
| 92h | **Maximum Memory Range Sets Exceeded:** The command exceeded the maximum number of Memory Range sets supported by the namespace. |
| 95h | **Memory Range Set In Use:** The command references a Memory Range Set ID is being used by an Execute Program command that has not yet completed. |
| 97h | **Overlapping Memory Ranges:** The command specified overlapping Memory Ranges. |

### 4.2.3 Program Activation Management command

The Program Activation Management command is used by the host to manage program activation. The command uses Command Dword 10. All other Command Dwords are reserved.

Program activation is managed on a per-namespace basis.

Program activation is not persistent, and shall be cleared on power on resets.

The Program Index (PIND) field is used as follows for individual operations:

- Activate: The PIND field is used to indicate the index of the program to be activated. If the specified index does not contain a program, the controller shall abort the with a status of No Program. Activate of an active program shall complete successfully.
- Deactivate: The PIND field is used to indicate the index of the program to be deactivated. Specifying a value of FFFFh is used to deactivate all programs in the namespace. If the value of FFFFh is specified and there are zero activated programs, the command shall complete successfully. Deactivate of an inactive program shall complete successfully.

**Figure 52: Program Activation Management – Command Dword 10**

| Bits | Description |
|---|---|
| 31:20 | Reserved |
| 19:16 | **Select (SEL):** This field selects the type of management operation to perform.<br><br>| Value | Description |<br>|---|---|<br>| 0h | Deactivate |<br>| 1h | Activate |<br>| 2h to Fh | Reserved | |
| 15:00 | **Program Index (PIND):** This field specifies the index of the program that this command applies to.<br><br>If the SEL field is cleared to 0h, a value of FFFFh in this field shall deactivate all programs for the specified namespace. If the SEL field is set to 1h and this field is set to FFFFh, then the controller shall abort the command with a status code of Invalid Program Index. |

### 4.2.3.1 Command completion

Upon completion of the Activate Program command, the controller posts a completion queue entry to the Admin Completion Queue. A command that attempts to activate a program that is already activated shall return success. A command that attempts to deactivate a program that is not activated shall return success.

**Figure 53: Program Activation Management – Command Specific Status Values**

| Value | Description |
|---|---|
| 8Eh | **Invalid Program Data:** Program data is not valid. |
| 8Ch | **Invalid Program Index:** Specified index is not a valid value. |
| 93h | **Maximum Programs Activated:** Cannot perform the activate operation since the namespace already has the maximum number of programs activated. |
| 96h | **No Program:** There is no program associated with the specified index. |
| 99h | **Program In Use:** Cannot perform the deactivate operation since a command referencing this program is in progress. |

# 5   Extended Capabilities

This I/O Command Set does not add any Extended Capabilities beyond what is defined in the NVM Express Base Specification.