Emulex® NVMe over Fibre Channel

User Guide
Release 12.0

Table of Contents

Chapter 1: Overview

This user guide provides instructions for installing and configuring NVMe over FC drivers on Emulex® LPe31000-series and LPe32000-series adapters in initiator and target systems, and on LPe35000-series adapters in target systems.

NVMe over FC is a relatively new protocol for solid-state storage devices built with nonvolatile memory technologies. NVMe provides substantially lower latency for storage I/O operations and significantly higher IOPs per device. NVMe scales up the number of devices it can address by adopting NVMe over fabric technology. LPe31000-series, LPe32000-series, and LPe35000-series HBAs are enabled for NVMe over fabrics. NVMe-enabled HBAs support NVMe over fabrics and SCSI concurrently, allowing data centers to transition to all-flash storage at their own pace.

You will need a basic understanding of NVMe over FC before configuring it. You can use a search engine to find various websites and books that describe NVMe over FC. In particular, you will need to understand the concept of *namespaces* (the NVMe equivalent of SCSI LUNs) and NVMe *subsystems* (containers of SCSI LUN equivalents).

In the case of Windows, the operating system lacks native FC support, so an FCP target is presented as a SCSI target. Similarly, an NVMe over FC subsystem is presented as a SCSI target, and a namespace discovered within an NVMe subsystem is presented as a LUN.

Emulex provides the following NVMe-related files on the Broadcom® website (www.broadcom.com):

- For Windows Server 2016, Windows Server 2012 SP2, and Windows Server 2012, the FC driver for Windows in the `elxdrvr-fc-<version>.exe` driver package supports both SCSI and NVMe on initiator servers.
- For SLES 12 SP3 and SLES 15, out-of-box NVMe drivers are available in the `elx-lpfc-dd-sles<distro>sp-<driver_version>-ds-1.tar.gz` driver package.

NOTE: Helper script files are no longer supported.

For SLES 12 SP3 and SLES 15, this document describes how to install the out-of-box drivers and how to configure the drivers using native NVMe CLI commands, which are part of the SLES 12 SP3 and SLES 15 operating systems.

For Windows, this document describes how to:

- Enable and configure NVMe on the initiator using the OneCommand® Manager GUI or OneCommand Manager CLI.
- Manage the NVMe target device using the OneCommand Manager CLI.

NOTE: The *Emulex NVMe over Fibre Channel for Windows Release Notes* provide interoperability information pertaining to the Windows initiator drivers. Refer to this document before installing the Windows NVMe over FC drivers.

1.1 Abbreviations

Acronym/Abbreviation	Description
CLI	command line interface
DID	device ID
DPS	
FC	Fibre Channel
FCP	Fibre Channel Protocol
GA	general availability
HBA	host bus adapter

Acronym/Abbreviation	Description
I/O	input/output
ID	identifier
IOPs	input/output operations per second
IOQ	input/output queue
LBA	logical block address
LUN	logical unit number
NQN	NVMe qualified name
NVMe	Nonvolatile Memory Express
NVMeT	Nonvolatile Memory Express target
SCSI	Small Computer System Interface
SLES	SUSE Linux Enterprise Server
WWNN	World Wide Node Name
WWPN	World Wide Port Name
XRI	extensible resource indicator

Chapter 2: Installing NVMe over FC

For SLES 12 SP3 and SLES 15, this chapter describes how to install the out-of-box NVMe drivers, and it describes the NVMe-specific driver parameters.

For Windows Server 2016, Windows Server 2012 SP2, and Windows Server 2012:

- The FC driver for Windows supports both SCSI and NVMe on initiator servers. Refer to the *Emulex Drivers for Windows for LightPulse Adapters User Guide* for installation instructions.
- You must install the OneCommand® Manager GUI to enable the NVMe driver on initiator servers. Refer to the *Emulex OneCommand Manager Application for LightPulse Adapters User Guide* for installation instructions. Note that installing the OneCommand Manager GUI also installs the OneCommand Manager CLI, which you can use to configure and monitor NVMe targets.

2.1 Installing the SLES 12 SP3 or SLES 15 GA Operating System

Install the SLES 12 SP3 or SLES 15 GA operating system on the computer, following the instructions provided with the operating system. Installing the operating system automatically installs an inbox initiator and target NVMe driver. Installation also allows you to install the `nvme-cli` utility and the `nvmetcli` utility. The `nvme-cli` utility supports initiation of new NVMe over fabric connections, and it functions as a general utility that allows you to query and manipulate an NVMe device. The `nvmetcli` utility supports the configuration of a system to be an NVMe over fabric target device.

NOTE: At this time, Emulex supports NVME over FC on target devices on SLES 12 SP3 only.

During the installation, configure the following items:

- Select the system role of **Default System**.
- In **Installation Settings**, select the following features:
  - Install **NVMe CLI**.
  - Install **NVMeT CLI** (SLES 12 SP3 only).

NOTE: You must update to the latest SUSE kernel and the latest SUSE release of the NVMe CLI and NVMeT CLI packages.

2.2 Installing the Out-of-Box NVMe over FC Driver

After the operating system is installed, perform the following steps to install the out-of-box NVMe over FC driver:

1. Download version 12.0 or later of the driver kit for SLES 12 SP3 or SLES 15 from the Documents and Downloads area of www.broadcom.com.
2. Log in as `root` to a terminal, and untar the driver kit:
   `tar -zxvf elx-lpfc-dd-sles<`***`distro`***`>sp-<`***`driver_version`***`>-ds-1.tar.gz`
3. Change to the directory into which the driver kit was extracted:
   `cd elx-lpfc-dd-sles<`***`distro`***`>sp-<`***`driver_version`***`>/`
4. Run the `elx_lpfc_install.sh` script with the `-n` option to install the NVMe over FC driver:
   `./elx_lpfc_install.sh -n`
   After the `elx_lpfc_install.sh` script has finished running successfully, the NVMe over FC driver is installed.
5. Reboot the system.

2.3 NVMe-Specific Driver Parameters

NVMe-specific driver parameters are displayed in the following table. For information on how to set the parameters, refer to the *Emulex Drivers for Linux for LightPulse Adapters User Guide*.

Parameter	Description	sysfs Visible	Activation
`lpfc_nvme_io_channel`	Defines the number of I/O channels supported by the driver. The default value is 0. A value of 0 means the driver looks at the number of online CPUs in the system. The default value is strongly recommended.	Yes	Driver reload
`lpfc_nvmet_mrq`	Configures the allocation of multi-receive queues (NVMe target mode only). The default value is 0 (the driver sets the appropriate value). The maximum value is 16.	Yes	Driver reload
`nvme_info`	Provides the NVMe configuration status for the port (read only)	Yes	Dynamic
`lpfc_enable_fc4_type`	When enabled, this parameter defines the FC4 types that are supported. The possible values are: ■ 1 = Enable just FCP (default) ■ 3 = Enable both FCP and NVMe Supported values are 1 and 3. The default value is 1.	Yes	Driver reload
`lpfc_enable_nvmet`	Specifies the WWPN of the ports enabled for NVMe target.	No	Driver reload
`lpfc_xri_split`	When enabled, this parameter defines the division of XRI resources between SCSI and NVMe. This parameter is used only if the value of the `lpfc_enable_fc4_type` parameter is 3 (register both FCP and NVMe) and the port is not configured for NVMeT. Supported values are in percentages. The `lpfc_xri_split` value is the percentage of XRI resources allocated to the SCSI port. The remaining percentage of XRI resources is allocated to NVMe. The supported range is 10% to 90%. The default value is 50%.	Yes	Driver reload

Chapter 3: Configuring NVMe over FC on SLES 12 SP3 and SLES 15

To configure NVMe over FC, you must perform the following procedures in the indicated sequence:

1. Configure NVMe over FC on target systems (see Appendix A, Configuring NVMe over FC on a Target (SLES 12 SP3 Only)).
2. Configure NVMe over FC on initiator systems, as described in this chapter.

NOTE: Before configuring NVMe over FC using native NVMe CLI commands, ensure that you have installed the latest Emulex firmware for the LPe31000-series, LPe32000-series, and LPe35000-series adapters.

NOTE: The target system must be configured and up before you connect the host system to the target.

3.1 Configuring NVMe over FC on SLES 12 SP3 and SLES 15 Using Native CLI Commands

To configure NVMe over FC on an initiator and to connect the initiator to an NVMe over FC target, perform the following steps:

1. Navigate to the `/etc/modprobe.d` subdirectory, and create a file with the driver name `lpfc.conf`.
2. Insert the following line in the `lpfc.conf` file, and save the file:
   ```
   options lpfc lpfc_enable_fc4_type=3
   ```

NOTE: `lpfc_enable_fc4_type=3` allows both SCSI and NVMe protocols to be enabled on the port; by default, the port is an initiator for the SCSI protocol.

3. Regenerate the ramdisk for the server by typing the following command:
   ```
   #dracut --force
   ```
4. Reboot the system.

   After the system is rebooted, the configured protocols are enacted on the FC ports.

   The `nvmefc-connect.rpm` file, which is installed with the out-of-box Linux driver, automatically causes the initiator to connect to the target devices. Connections are established whenever any of the following events occur, as long as the target is running:
   - A host system reboot
   - A linkup event on the initiator port
   - A fabric zone event

After configuring the initiator, you can run the `nvme_info` command to verify that your configuration is working by typing the following command on the initiator:

```
cat /sys/class/scsi_host/host<X>/nvme_info
```

where <*X*> is the host number.

Information similar to the following is displayed:

```
NVME Initiator Enabled
NVME LPORT lpfc0 WWPN x10000090fae39706 WWNN x20000090fae39706 DID x011000 ONLINE
NVME RPORT       WWPN x10000090fa94277a WWNN x20000090fa94277a DID x011100 INITIATOR ONLINE
```

```
NVME RPORT       WWPN x10000090fa942779 WWNN x20000090fa942779 DID x011400 TARGET DISCSRVC ONLINE

NVME Statistics
LS: Xmt 0000000000000019 Cmpl 0000000000000019
FCP: Rd 000000000000002e Wr 0000000000000011 IO 000000000000000c
    Cmpl 000000000000004b Outstanding 0000000000000000
```

5. If multipathing is used in SLES 15, skip to Step 6. For SLES 12 SP3, or for SLES 15 with a single path, use the `nvme list` command to display target connections by typing the following command on the initiator:
   `nvme list`

NOTE: This document covers are two similar commands:

- `nvme list` (with a space) is a native NVMe CLI command.
- `nvme-list` (with a hyphen) is an Emulex OneCommand Manager CLI command.

This chapter refers to the `nvme list` command.

Information similar to the following is displayed:

```
Node          SN                Model  Namespace  Usage                    Format           FW Rev
------------- ----------------- ------ ---------- ------------------------ ---------------  ---------
/dev/nvme0n1  981fadb118adb0fa  Linux  1          268.44  GB / 268.44  GB  512   B +  0 B    4.4.70-2
/dev/nvme0n2  981fadb118adb0fa  Linux  2          268.44  GB / 268.44  GB  512   B +  0 B    4.4.70-2
```

Skip to Step 7.

6. For multipathing in SLES 15, type the following command to display target connections on the initiator:
   `nvme list-subsys`
   Information similar to the following is displayed:

```
nvme-subsys0 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:19d48a6d-3644-4f65-8046-80bd88fe45a6
\
 +- nvme0 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f1f:pn-
0x100000109b346f1f
 +- nvme3 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f20:pn-
0x100000109b346f20
nvme-subsys1 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:2c6d862f-b77e-43b1-b345-7f990125088e
\
 +- nvme1 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f1f:pn-
0x100000109b346f1f
 +- nvme6 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f20:pn-
0x100000109b346f20
nvme-subsys2 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:4bbbd877-6de2-4a67-ad56-e0333e2437b5
\
 +- nvme2 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f1f:pn-
0x100000109b346f1f
 +- nvme7 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f20:pn-
0x100000109b346f20
nvme-subsys3 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:c8da7263-fefe-4563-a14c-a9f0362dcb03
\
 +- nvme5 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f1f:pn-
0x100000109b346f1f
 +- nvme9 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f20:pn-
0x100000109b346f20
nvme-subsys4 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:5f1d7788-fab1-45cf-af9c-d30dc9a9ce89
\
```

```
 +- nvme4 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f1f:pn-
0x100000109b346f1f
 +- nvme8 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f20:pn-
0x100000109b346f20
```

NOTE: For additional information on multipathing, see Appendix B, Multipathing.

7. You can run the `lsblk` command on the initiator to display all the block devices, including newly discovered ones, by typing the following command:

   `lsblk`

   Information similar to the following is displayed:

```
 NAME                                              MAJ:MIN  RM   SIZE      RO   TYPE     MOUNTPOINT
sda                                                  8:0    0   136.7G    0    disk
├─sda1                                               8:1    0   2G        0    part
├─sda2                                               8:2    0   40G       0    part
├─sda3                                               8:3    0   94.7G     0    part
└─3600508b1001037383941424344450400                254:0    0   136.7G    0    mpath
  ├─3600508b1001037383941424344450400-part1        254:1    0   2G        0    part     [SWAP]
  ├─3600508b1001037383941424344450400-part2        254:2    0   40G       0    part     /var/opt
  └─3600508b1001037383941424344450400-part3        254:3    0   94.7G     0    part     /home
sdb                                                  8:16   0   3G        0    disk
  └─3600000e00d2a0000002a017300340000                254:4    0   3G        0    mpath
sdc                                                  8:32   0   3G        0    disk
  └─3600000e00d2a0000002a017300350000                254:5    0   3G        0    mpath
sdd                                                  8:48   0   3G        0    disk
  └─3600000e00d2a0000002a017300360000                254:6    0   3G        0    mpath
nvme0n1                                            259:0    0   3G        0    disk
nvme0n2                                            259:1    0   3G        0    disk
```

   Note that the `/dev/sde` and `/dev/sdf` block devices, which were previously referred to on the NVMe target system, are now discovered by the NVMe initiator system, and they appear as the NVMe block devices `/dev/nvme0n1` and `/dev/nvme0n2`.

3.2 Emulex Support for NVMe CLI Commands

The SLES 12 SP3 and SLES 15 operating systems include many native NVMe CLI commands for initiators. Of these, Emulex has tested, and thus supports, only the following commands at this time:

- `list`
- `connect-all`
- `connect`
- `gen-hostnqn`
- `list-subsystem` (SLES 15 only)

Documentation for these commands is available with the operating system.

Chapter 4: Enabling and Configuring NVMe over FC on Windows

This chapter provides instructions for the following tasks:

- Enabling and configuring NVMe over FC on a Windows initiator using the OneCommand Manager GUI or the OneCommand Manager CLI
- Monitoring NVMe targets using the OneCommand Manager CLI

NOTE: Windows initiator systems must connect to target devices that are configured for NVMe.

4.1 Enabling and Configuring NVMe over FC on an Initiator

You can configure NVMe driver parameters using the OneCommand Manager GUI or the OneCommand Manager CLI.

- Refer to the *Emulex OneCommand Manager Application for LightPulse Adapters User Guide* for detailed information on starting the OneCommand Manager GUI and navigating to the **Driver Parameters** tab.
- Refer to the *Emulex OneCommand Manager Command Line Interface for LightPulse Adapters User Guide* for detailed information on using the `SetDriverParam` command.

The following table shows the NVMe driver parameters and supplements the information in Table 1 of the *Emulex Drivers for Windows for LightPulse Adapters User Guide*, which provides specific information about driver parameters.

Parameter	Definitions	Activation Requirement	Notes
`EnableNVMe`	`EnableNVMe` enables or disables driver NVMe over FC functionality. ■ `0` = Disable NVMe ■ `1` = Enable NVMe Value: 0 or 1 Default = 0	Reboot	Enabling NVME causes the driver to allocate more memory resources to support NVMe over FC. It also activates NVMe discovery in addition to FCP discovery, which allows the driver to discover both FCP targets and NVMe over FC targets. The driver runs in either FCP mode or in FCP+NVMe mode. NVMe-only mode is not supported.
`NumNVMENode`	`NumNVMENode` specifies the maximum number of NVMe nodes supported by the driver. An NVMe node is either an NVMe discovery service or an NVMe subsystem. Value: 8 to 512 (decimal) or 0x8 to 0x200 (hexadecimal) Default = 16	Reboot	
`NumNVMENS`	`NumNVMENS` specifies the maximum number of NVMe namespaces supported by the driver within an NVMe subsystem. Value: 8 to 255 (decimal) or 0x8 to 0xFF (hexadecimal) Default = 8	Reboot	

Parameter	Definitions	Activation Requirement	Notes
NumNVMEIOQ	NumNVMEIOQ specifies the maximum number of NVMe I/O queues supported by the driver within an NVMe subsystem. Value: 4 to 255 (decimal) or 0x4 to 0xFF (hexadecimal)	Reboot	The NumNVMEIOQ parameter caps the maximum number of IOQs supported by the driver. The driver allocates resources based on this setting. However, the following conditions apply: ■ The number of IOQs created on the subsystem also depends on the maximum number of IOQs that the subsystem supports. ■ The number of IOQs created is the minimum value of NumNVMEIOQ and the maximum number of IOQs supported by the subsystem. For best performance and to avoid wasting resources, verify the subsystem capabilities before setting this value. Issue the nvme-get-feature command, with the FeatureID parameter set to 0x7, to read the number of supported queues.
NVMEKATimeInt	NVMEKATimeInt specifies the number of 5-second time intervals that the NVMe Keep Alive command is sent for each NVMe subsystem association. The Keep Alive timeout is set to 3 times the value of NVMEKATimeInt, in seconds. Value: 1 to 12 Default = 3	Reboot	For example, if NVMEKATimeInt is set to 2: ■ The Keep Alive command is sent every 10 seconds (2 × 5). ■ The Keep Alive timeout is set to 30 seconds (10 × 3).

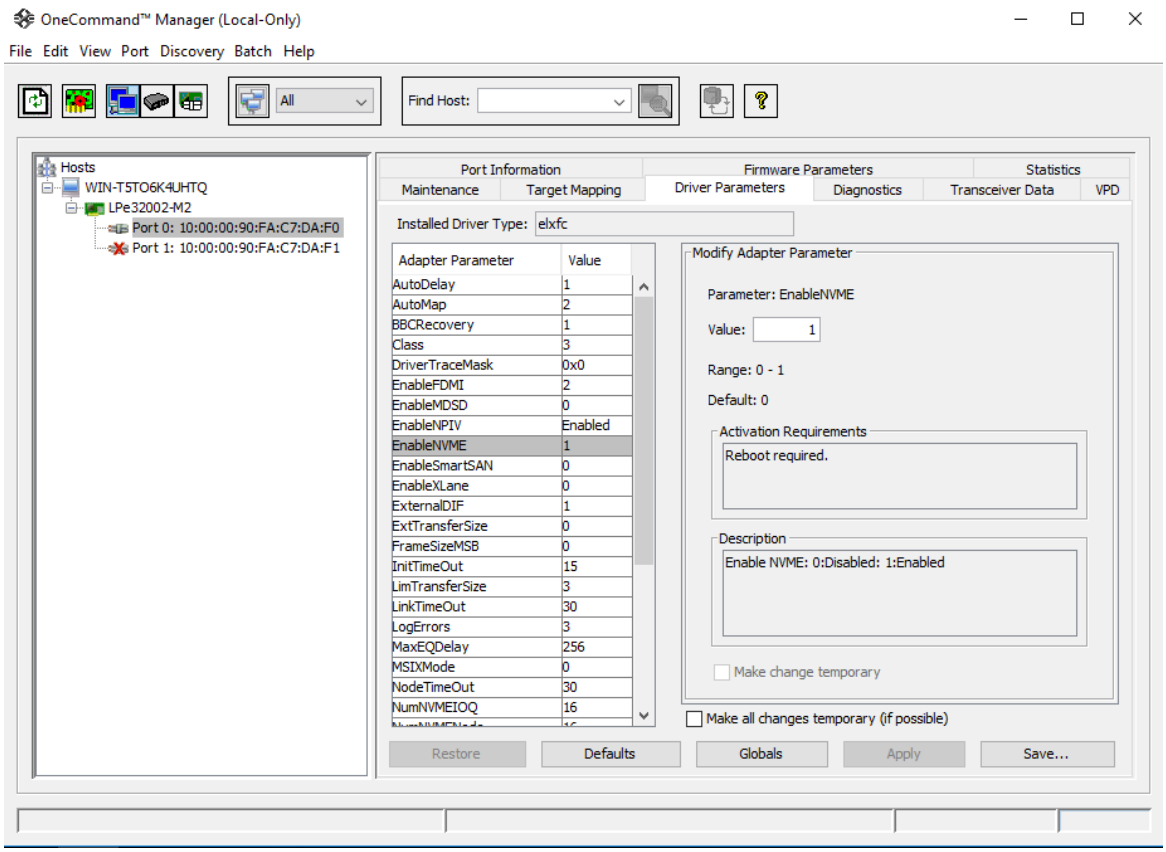
NOTE: Using the OneCommand Manager CLI, you can use the GetDriverParam command to view the current settings for the parameters.

Two existing driver parameters are affected by the NVMe over FC implementation:

- AutoMap – If the EnableNVME parameter is set to 1, the driver automatically sets the AutoMap parameter to the default value of 2, automap by WWPN.
- EnableXLane – If the EnableXLane parameter is set to 1, ExpressLane™ functionality is applied only to FCP LUNs, and not to NVMe namespaces.

The following figure shows an example of setting the EnableNVME parameter.

Figure 1: Enabling NVMe over FC



4.1.1 Driver Limitations

The following limitations apply to the Windows driver when it is enabled for NVMe over FC:

- NVMe subsystems are assigned SCSI path IDs and target IDs, and they do not persist across boots. Thus, persistent binding is not supported.
- NVMe namespaces are assigned LUN numbers in the order in which they are discovered, and they do not persist across boot. Thus, LUN mapping and LUN masking are not supported.
- The driver reads only the first 35 discovery log entries, and it does not support referral depth. Thus, the maximum number of subsystems on an FC node is limited to 35.
- The following features are not supported:
  - Crash dump
  - ExpressLane (however, support is available on FCP LUNs)

4.1.2 Error Codes

The following NVMe error codes have been added to the driver for Windows.

Byte 0x10	Interpretation
0x80 (severe)	The NVMe over FC Keep Alive feature failed on the indicated path and target.
0x81 (command)	The indicated IOQ is not available.
0x82 (malfunction)	IOQ creation failure. Byte 0x11 indicates the number of IOQs created. Byte 0x12 indicates the expected number of IOQs to be created.

4.2 Monitoring NVMe Targets

Some existing OneCommand Manager CLI commands were updated to support NVMe targets, and some new NVMe-specific commands were added.

For detailed information about the OneCommand Manager CLI commands, refer to the *Emulex OneCommand Manager Command Line Interface for LightPulse Adapters User Guide*. The information in this section is intended to supplement the user guide, not replace it.

4.2.1 Updated CLI Commands

4.2.1.1 The Help Command

The `NVMe` functional group has been added to the `Help` command. The following commands belong to the `NVMe` group:

- `nvme-attach-ns`
- `nvme-create-ns`
- `nvme-delete-ns`
- `nvme-detach-ns`
- `nvme-get-sstable`
- `nvme-get-feature`
- `nvme-id-ctrl`
- `nvme-id-ns`
- `nvme-list`
- `nvme-list-ctrl`
- `nvme-list-ns`
- `nvme-set-sstable`

For information about the commands in the `NVMe` group, see Section 4.2.2, NVMe-Specific Commands.

4.2.1.2 The ListHBAs Command

The `ListHBAs` command includes a new value for port type that indicates whether a port is an FC initiator or an FC+NVMe initiator.

Syntax

`ListHBAs [local] [m=`*`model`*`] [pt=`*`type`*`] [down]`

Parameters

`local`	Displays only local adapters.
`m=`*`model`*	The model filter. Append `*` to the end of the model name for a wildcard match. For example, `LPe32*`.
`pt=`*`type`*	The port type filter. Valid types are `FC` and `NVMe`.
`down`	Not applicable.

Example

```
hbacmd listhbas pt=nvme
```

Information similar to the following is displayed:

```
Manageable HBA List

Port WWN       : 10:00:00:90:fa:94:23:c6
Node WWN       : 20:00:00:90:fa:94:23:c6
Fabric Name    : 00:00:00:00:00:00:00:00
Flags          : 8000e300
Host Name      : APPDEV100
Mfg            : Emulex Corporation
Serial No.     : TSN-0090fa9423c6
Port Number    : 0
Mode           : Initiator
PCI Bus Number : 36
PCI Function   : 0
Port Type      : FC+NVMe
Model          : LPe32002-M2

Port WWN       : 10:00:00:90:fa:94:23:c7
Node WWN       : 20:00:00:90:fa:94:23:c7
Fabric Name    : 00:00:00:00:00:00:00:00
Flags          : 8000e300
Host Name      : APPDEV100
Mfg            : Emulex Corporation
Serial No.     : TSN-0090fa9423c6
Port Number    : 1
Mode           : Initiator
PCI Bus Number : 36
PCI Function   : 1
Port Type      : FC+NVMe
Model          : LPe32002-M2
```

4.2.1.3 The AllNodeInfo Command

A new parameter has been added to the `AllNodeInfo` command to filter the display of nodes to SCSI (FCP)-only or to NVMe-only nodes. If a filter option is not specified in the command line, the output lists both SCSI and NVMe nodes.

Syntax

```
hbacmd AllNodeInfo <WWPN> [S|N]
```

Parameters

WWPN	The WWPN of an FC function.
Filter Type	The type of filter to apply (optional). S = Display SCSI nodes only. N = Display NVMe nodes only.

Examples

This example shows the NVMe node selected:

```
hbacmd AllNodeInfo 10:00:00:90:fa:5d:05:a9 N
```

Information similar to the following is displayed:

```
Target Type    : NVMe
Node Type      : READY
FCP ID         : 20400
SCSI Bus Number: 0
SCSI Target Num: 0   // Not Applicable for Linux NVMe nodes
Node WWN       : 50:06:01:60:90:20:5C:38
Port WWN       : 50:06:01:60:10:20:5C:38
OS Device Name : \\.\Scsi5:0:0
```

This example shows the SCSI node selected:

```
hbacmd AllNodeInfo 10:00:00:90:fa:5d:05:a9 S
```

Information similar to the following is displayed:

```
Target Type    : SCSI
Node Type      : READY
FCP ID         : 20400
SCSI Bus Number: 0
SCSI Target Num: 1
Node WWN       : 50:06:01:60:90:20:5C:38
Port WWN       : 50:06:01:60:10:20:5C:38
OS Device Name : \\.\Scsi5:0:1
```

This example shows both the NVMe node and the SCSI node selected:

```
hbacmd AllNodeInfo 10:00:00:90:fa:5d:05:a9
```

Information similar to the following is displayed:

```
Target Type    : NVMe
Node Type      : READY
FCP ID         : 20400
SCSI Bus Number: 0
SCSI Target Num: 0
Node WWN       : 50:06:01:60:90:20:5C:38
Port WWN       : 50:06:01:60:10:20:5C:38
OS Device Name : \\.\Scsi5:0:0
SCSI Target Num: 0
Node WWN       : 50:06:01:60:90:20:5C:38
Port WWN       : 50:06:01:60:10:20:5C:38
OS Device Name : \\.\Scsi5:0:0
Target Type    : SCSI
Node Type      : READY
FCP ID         : 20400
SCSI Bus Number: 0
SCSI Target Num: 1
Node WWN       : 50:06:01:60:90:20:5C:38
```

```
Port WWN       : 50:06:01:60:10:20:5C:38
OS Device Name : \\.\Scsi5:0:1
```

4.2.2 NVMe-Specific Commands

The OneCommand Manager CLI commands in this section are specifically used for NVMe over FC targets.

Support by NVMe targets for some of the commands in this section is optional and depends on whether namespace management is supported by the NVMe target. See the command descriptions for support requirements.

To determine whether namespace management is supported by the target, issue the `nvme-id-ctrl` command (see Section 4.2.2.5, The nvme-id-ctrl Command) and examine the value displayed in the `Namespace Management` field.

Some commands require you to enter parameters, such as the `Target WWPN`, `NQN`, and `CtrlID`. To determine these parameters, issue the `nvme-list` command (see Section 4.2.2.3, The nvme-list Command).

NOTE: In the `nvme-list` command output, the `Target WWPN` is referred to as the *Port WWN*.

NOTE: This document covers are two similar commands:

- `nvme list` (with a space) is a native NVMe CLI command.
- `nvme-list` (with a hyphen) is an Emulex OneCommand Manager CLI command.

This chapter refers to the `nvme-list` command.

4.2.2.1 The nvme-set-sstable Command

The `nvme-set-sstable` command allows you to update the NVMe subsystem management table that is maintained for each FC+NVMe port. If an NVMe target does not support the NVMe discovery service, subsystems on that target are not present in the discovered subsystem list returned by the `nvme-list` command. Subsystems that are not discoverable by the NVMe discovery service must be added to a subsystem management table that is maintained for each FC+NVMe port. The `nvme-set-sstable` command allows you to manage the contents of the subsystem management table.

NOTE: The subsystem management table can contain up to 16 entries.

Syntax

`hbacmd nvme-set-sstable <WWPN> <Target WWPN> <Target WWNN> <NQN> <CtrlID> <Action> [SQSize]`

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device.
Target WWNN	The WWNN of an FC port on an NVMe-capable device.
NQN	The qualified name of the NVMe subsystem.
CtrlID	The controller identifier, in hexadecimal.
Action	The action to be performed: `1` = Add the subsystem to the management table. `2` = Remove the subsystem from the management table.
SQSize	The size of the Admin Submission Queue for the specified controller. The range of valid values is 2 to 4096. This parameter is required if `Action = 1`; otherwise it is optional.

Support Requirement

- Mandatory

Example

```
hbacmd nvme-set-sstable 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x0001 1 4096
```

Information similar to the following is displayed:

```
NVMe Subsystem table successfully updated for port 10:00:00:90:fa:5d:05:a9
```

4.2.2.2 The nvme-get-sstable Command

The `nvme-get-sstable` command allows you to retrieve the NVMe subsystem management table that is maintained for each FC+NVMe port.

Syntax

```
hbacmd nvme-get-sstable <WWPN> [Target WWPN]
```

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device (optional).

Support Requirement

- Mandatory

Example

```
hbacmd nvme-get-sstable 10:00:00:90:fa:5d:05:a9
```

Information similar to the following is displayed:

```
NVMe Qualified Name : nqn.2014-08.com.example1:nvme.host.sys.xyz
Port WWN            : 50:06:01:60:90:20:5C:38
Node WWN            : 50:06:01:60:10:20:5C:38
CtrlID              : 0X0001

NVMe Qualified Name : nqn.2014-08.com.example2:nvme.host.sys.xyz
Port WWN            : 50:06:01:60:90:20:5C:39
Node WWN            : 50:06:01:60:10:20:5C:39
CtrlID              : 0X0001
```

If an invalid *Target WWPN* is used, the following error message is displayed:

```
ERROR: The Subsystem table does not contain any entries for the specified initiator port.
ERROR: <823>: No subsystem entries were found for the specified initiator port.
```

4.2.2.3 The nvme-list Command

The `nvme-list` command lists the NVMe over FC subsystems recognized by the FC driver connected to the FC port. The list contains all discovered subsystems, as well as those that were added to the subsystem table using the `nvme-set-sstable` command.

Syntax

```
hbacmd nvme-list <WWPN> [Target WWPN]
```

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device (optional).

Support Requirement

- Mandatory

Example

```
hbacmd nvme-list 10:00:00:90:fa:5d:05:a9
```

Information similar to the following is displayed:

```
Discovered NVMe Subsystems for 10:00:00:90:fa:5d:05:a9
NVMe Qualified Name : nqn.2014-08.com.example:nvme.host.sys.xyz
Port WWN            : 50:06:01:60:90:20:5C:38
Controller ID       : 0x0001
Model Number        : NVMfxx999
Serial Number       : 0123456789
Firmware Version    : 1.2.0.1
Total Capacity      : 1 TB
Unallocated Capacity: 500 GB

Configured NVMe Subsystems for 10:00:00:90:fa:5d:05:a9
NVMe Qualified Name : nqn.2014-08.com.example:nvme.host.m32
Port WWN            : 30:16:01:62:90:21:CC:41
Controller ID       : 0x0001
Model Number        : NVMf0103
Serial Number       : 0123456789
Firmware Version    : 1.2.0.5
Total Capacity      : 2 TB
Unallocated Capacity: 400 GB
```

4.2.2.4 The nvme-list-ctrl Command

The `nvme-list-ctrl` command displays the controllers for a specified NVMe target.

Syntax

```
hbacmd nvme-list-ctrl <WWPN> <Target WWPN> <NQN> [NSID]
```

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device.
NQN	The qualified name of the NVMe subsystem.
NSID	The namespace identifier (optional), in hexadecimal. Use it to list only controllers attached to the specified namespace.

NOTE: Only the controllers on the subsystem specified in the command line are displayed. Controllers for other subsystems are not displayed, even if they reside within the same NVME target WWPN.

Support Requirement

- Optional

Example

```
hbacmd nvme-list-ctrl 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz
```

Information similar to the following is displayed:

```
Controllers for subsystem nqn.2014-08.com.example:nvme.host.sys.xyz

Number of Controllers: 3

Controller ID: 0x0000
Controller ID: 0x0001
Controller ID: 0x0002
```

4.2.2.5 The nvme-id-ctrl Command

The `nvme-id-ctrl` command sends the NVMe `identify` command for a controller and displays the results.

Syntax

`hbacmd nvme-id-ctrl <WWPN> <Target WWPN> <NQN> [CtrlID=<ctrlid>] [raw]`

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device.
NQN	The qualified name of the NVMe subsystem.
ctrlid	The controller that processes the request (optional), in hexadecimal.
`raw`	Specifies that the output is to be provided in hexadecimal format (optional).

NOTE: If the `raw` parameter is not specified, the output is provided as a short list of controller attributes in human-readable format. If the `raw` parameter is specified, the entire data structure is displayed in raw (hexadecimal) format, including the vendor-specific data bytes. See the examples in this section.

Support Requirement

- Mandatory

Examples

This example shows no optional parameter specified:

```
hbacmd nvme-id-ctrl 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz
```

Information similar to the following is displayed:

```
Attributes for Controller 0x0001:

IEEE OUI Identifier                   : 032957
Maximum data transfer size            : 5120
NVMe version supported                : 1.2.1
Number of power states supported      : 0
Warning temperature level             : 140
Critical temperature level            : 160
Maximum Number of namespaces          : 3
Submission Queue Size                 : Min 64, Max 64
Completion Queue Size                 : Min 16, Max 16
Maximum commands supported at one time: 2048
Atomic write unit normal              : 0
Atomic write unit power fail          : 0
```

```
Controller Features:
    Namespace Management              : Supported
    Non-zero feature Select option    : Not Supported
```

This example shows the `raw` parameter specified:

```
hbacmd nvme-id-ctrl 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz raw
```

Information similar to the following is displayed:

```
Raw data for Controller 0x0001:

Standard data bytes:

Byte Dump
0000: 03 04 07 00 00 00 00 60 40 0c 70 06 8c 0a 00 00
0010: 04 02 00 0a 45 4d 55 4c 45 58 20 20 20 20 20 20
0020: 20 20 20 20 00 00 17 6a 41 46 42 52 2d 35 37 46
0030: 35 4d 5a 2d 45 4c 58 20 20 20 20 20 52 03 00 48
0040: 00 3a 00 00 41 41 31 32 32 31 4a 30 43 32 50 20
0050: 20 20 20 20 31 32 30 35 32 38 20 20 68 fa 05 3a
0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
……: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
……: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
……: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0BF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Vendor-specific data bytes:
Byte Dump
0C00: 03 04 07 00 00 00 00 60 40 0c 70 06 8c 0a 00 00
0C10: 04 02 00 0a 45 4d 55 4c 45 58 20 20 20 20 20 20
0C20: 20 20 20 20 00 00 17 6a 41 46 42 52 2d 35 37 46
0C30: 35 4d 5a 2d 45 4c 58 20 20 20 20 20 52 03 00 48
0C40: 00 3a 00 00 41 41 31 32 32 31 4a 30 43 32 50 20
0C50: 20 20 20 20 31 32 30 35 32 38 20 20 68 fa 05 3a
0C60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0C70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0C80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0C90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0CA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0CB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
……: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
……: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
……: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0FF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

This example shows the `CtrlID` parameter specified:

```
hbacmd nvme-id-ctrl 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz CtrlID=0x0001
```

Information similar to the following is displayed:

```
Attributes for Controller 0x0001:

IEEE OUI Identifier                   : 032957
Maximum data transfer size            : 5120
NVMe version supported                : 1.2.1
Number of power states supported      : 0
Warning temperature level             : 140
Critical temperature level            : 160
Number of namespaces                  : 3
Submission Queue Size                 : Min 128, Max 128
Completion Queue Size                 : Min 16, Max 16
Maximum commands supported at one time: 2048
Atomic write unit normal              : 0
Atomic write unit power fail          : 0

Controller Features:
    Namespace Management              : Supported
```

If an invalid `CtrlID` parameter is used, the following error message is displayed:

```
ERROR: HBACMD_NvmeIdCtrl: RM_GetNvmeControllerAttributes call failed (821)
ERROR: <821>: The specified subsystem was not found.
```

4.2.2.6 The nvme-get-feature Command

The `nvme-get-feature` command submits an NVMe `get-feature` command and displays the applicable results. The results might be the value of the feature, or they might include a feature structure if the feature requires it. Currently, only the following features are known to return a data structure:

- LBA range type
- Autonomous power state transition
- Host memory buffer

Syntax

```
hbacmd nvme-get-feature <WWPN> <Target WWPN> <NQN> <FeatureID> [CtrlID=<ctrlid>][raw]
[Select=<select>] [NSID=<nsid>] [vector=<vector>] [tempSelect=<tempselect>]
```

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device.
NQN	The qualified name of the NVMe subsystem.

FeatureID The feature identifier:

0x1 = Arbitration
0x2 = Power management
0x3 = LBA range type
0x4 = Temperature threshold
0x5 = Error recovery
0x6 = Volatile write cache
0x7 = Number of queues
0x8 = Interrupt coalescing (not supported)
0x9 = Interrupt vector configuration (not supported)
0xA = Write atomicity normal
0xB = Asynchronous event configuration
0xC = Autonomous power state transition
0xD = Host memory buffer
0xF = Keep alive timer
0x80 = Software progress marker (optional feature)
0x81 = Host identifier (mandatory feature)
0x82 = Reservation notification mask (optional feature)
0x83 = Reservation persistence (optional feature)

ctrlid The controller that processes the request (optional), in hexadecimal.

raw Specifies that the output is to be provided in hexadecimal format if the feature returns a data structure; ignored if no data structure is returned (optional).

select The attribute value to be returned (optional):

0 = Current
1 = Default
2 = Saved
3 = Supported capabilities:

- Savable: You can change the value, and the value persists after a controller reset. If a value is not savable, you can change the value, but the value will not persist after a controller reset.
- Changeable: You can save the feature using the SetFeature command (not currently supported).
- Namespace specific: The *FeatureID* is specific to the namespace identified in the command line by *nsid*.

If no Select parameter is specified, output contains the current, default, and saved values for the selected feature and the supported capabilities of the feature.

Some Select parameter attribute values may not be available on all targets. To determine the level of support for the Select parameter, issue the nvme-id-ctrl command and examine the value displayed for Non-zero feature Select option. If the value is Not Supported, the only allowed value for the Select parameter is 0 (current). An attempt to enter a nonzero value for the Select parameter results in an Invalid Argument error. Additionally, if no Select parameter is specified, only the current value for the selected feature is presented.

nsid The namespace for which a feature is to be retrieved.

vector The interrupt vector for which the configuration settings are to be retrieved, in hexadecimal. Valid values are 0x0 to 0xffff. Applies only to the interrupt vector configuration feature (*FeatureID*=0x9).

tempselect The temperature threshold value to be retrieved:
`0x00` = Over-temperature threshold value for composite temperature
`0x01` = Over-temperature threshold value for temperature sensor 1
`0x02` = Over-temperature threshold value for temperature sensor 2
...
`0x08` = Over-temperature threshold value for temperature sensor 8
`0x10` = Under-temperature threshold value for composite temperature
`0x11` = Under-temperature threshold value for temperature sensor 1
`0x12` = Under-temperature threshold value for temperature sensor 2
...
`0x18` = Under-temperature threshold value for temperature sensor 8

Support Requirements

- Mandatory:
  - 0x1 = Arbitration
  - 0x2 = Power management
  - 0x4 = Temperature threshold
  - 0x5 = Error recovery
  - 0x7 = Number of queues
  - 0xA = Write atomicity normal
  - 0xB = Asynchronous event configuration
  - 0x81 = Host identifier
- Mandatory only if reservations are supported by namespace management; optional otherwise:
  - 0x82 = Reservation notification mask
  - 0x83 = Reservation persistence
- Optional:
  - 0x3 = LBA range type
  - 0x6 = Volatile write cache
  - 0xC = Autonomous power state transition
  - 0xD = Host memory buffer
  - 0xF = Keep alive timer
  - 0x80 = Software progress marker
- Not supported:
  - 0x8 = Interrupt coalescing
  - 0x9 = Interrupt vector configuration

NOTE: To determine whether namespace management supports reservations, issue the `nvme-id-ns` command (see Section 4.2.2.8, The nvme-id-ns Command), and examine the value displayed in the `Reservations` field.

NOTE: If the `raw` parameter is not specified, the output provides all of the data available for the specified feature, in human-readable format. The data includes all of the supported attribute values, such as `current`, `saved`, and `default`, as well as any other attribute values returned in an additional data structure.

Examples

This example shows no optional parameters specified:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x1
```

Information similar to the following is displayed:

```
Feature: Arbitration

Supported Capabilities
    Savable            : Yes
    Namespace-Specific : No
    Changeable         : Yes

Attribute Name           Current        Saved        Default
Arbitration Burst           32            32             64
Low Priority Weight        128           128            128
Medium Priority Weight      64            64             64
High Priority Weight        32            32             32
```

This example shows the arbitration feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x1 Select=0
```

Information similar to the following is displayed:

```
Feature: Arbitration

Attribute Name                Value(Current)
Arbitration Burst                     32
Low Priority Weight                  128
Medium Priority Weight                64
High Priority Weight                  32
```

This example shows the power management feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x2 Select=1
```

Information similar to the following is displayed:

```
Feature: Power Management

Attribute Name                Value(Default)
Power State                        3
Workload Hint                      1
```

This example shows the LBA range type feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x3 Select=2 raw
```

Information similar to the following is displayed:

```
Feature: LBA Range Type

Byte Dump (Saved Values)
0000: 03 04 07 00 00 00 00 60 40 0c 70 06 8c 0a 00 00
0010: 04 02 00 0a 45 4d 55 4c 45 58 20 20 20 20 20 20
0020: 20 20 20 20 00 00 17 6a 41 46 42 52 2d 35 37 46
0030: 35 4d 5a 2d 45 4c 58 20 20 20 20 20 52 03 00 48
0040: 00 3a 00 00 41 41 31 32 32 31 4a 30 43 32 50 20
```

This example shows the temperature threshold feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x4 Select=0 tempSelect=0x01
```

Information similar to the following is displayed:

```
Feature: Temperature Threshold

Threshold Type: Over Temperature Threshold
Temperature Select: Sensor 1

Attribute Name                  Value(Current)
Temperature Threshold            255
```

This example shows the error recovery feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x5 Select=0
```

Information similar to the following is displayed:

```
Feature: Error Recovery

Attribute Name                              Value(Current)
Time Limited Error Recovery                    510
Deallocated Logical Block Error Enable         Enabled
```

This example shows the volatile write cache feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x6 Select=0
```

Information similar to the following is displayed:

```
Feature: Volatile Write Cache

Attribute Name                              Value(Current)
Volatile Write Cache Enable                    Disabled
```

This example shows number of queues feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x7 Select=0
```

Information similar to the following is displayed:

```
Feature: Number of Queues

Attribute Name                              Value(Current)
Number of I/O Submission Queues Allocated     63
Number of I/O Completion Queues Allocated     63

Note: The above values are zero based
```

This example shows the interrupt coalescing feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x8 Select=0
```

Information similar to the following is displayed:

```
Feature: Interrupt Coalescing

Attribute Name                              Value(Current)
Aggregation Threshold                           1024
Aggregation Time (ms)                           2048
```

This example shows the interrupt vector configuration feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x9 0 --vector=0x1
```

Information similar to the following is displayed:

```
Feature: Interrrupt Vector Configuration

Attribute Name                              Value(Current)
Interrupt Vector                              2047
Coalescing Disable                            Enabled
```

This example shows the write atomicity normal feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0xA Select=1
```

Information similar to the following is displayed:

```
Feature: Write Atomicity Normal

Attribute Name                              Value(Default)
Disable Normal                                 Disabled
```

This example shows the asynchronous event configuration feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0xB Select=2
```

Information similar to the following is displayed:

```
Feature: Asynchronous Event Configuration

Attribute Name                              Value(Saved)
SMART/Health Critical Warnings                Enabled
Namespace Attribute Notices                   Disabled
Firmware Activation Notices                   Enabled
```

This example shows the autonomous power state transition feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0xC Select=0
```

Information similar to the following is displayed:

```
Feature: Autonomous Power State Transition

Attribute Name                              Value(Current)
Autonomous Power State Transition Enable    Enabled
```

This example shows the host memory buffer feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0xD Select=0 raw
```

Information similar to the following is displayed:

```
Feature: Host Memory Buffer

Raw data for Dword 0
Byte Dump (Current Values)
0000: 03 00 00 00

Raw Data for Attributes Data Structure
Byte Dump (Current Values)
0000: 03 04 07 00 00 00 00 60 40 0c 70 06 8c 0a 00 00
0010: 04 02 00 0a 45 4d 55 4c 45 58 20 20 20 20 20 20
0020: 20 20 20 20 00 00 17 6a 41 46 42 52 2d 35 37 46
……: 35 4d 5a 2d 45 4c 58 20 20 20 20 20 52 03 00 48
0FF0: 00 3a 00 00 41 41 31 32 32 31 4a 30 43 32 50 20
```

Human-readable sample output:

```
Attribute Name                              Value(Current)
Enable Host Memory                              Enabled
Memory Return                                   Enabled
```

This example shows the keep alive timer feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0xF Select=0 raw
```

Information similar to the following is displayed:

```
Feature: Keep Alive Timer

Attribute Name                              Value(Current)
Keep Alive Timeout (ms)                      45000
```

This example shows the software progress marker feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x80 Select=0
```

Information similar to the following is displayed:

```
Feature: Software Progress Marker

Attribute Name                              Value(Current)
Pre-boot Software Load Count                      128
```

This example shows the host identifier feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x81 Select=0 raw
```

Information similar to the following is displayed:

```
Feature: Host Identifier

Byte Dump (Current Values)
0000: 03 00 00 00 03 00 00 00 03 00 00 00 03 00 00 00
```

Human-readable sample output:

```
Attribute Name                          Value(Current)
Host Identifier                         0xa7a5b1323132
```

This example shows the reservation notification mask feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x82 Select=0
```

Information similar to the following is displayed:

```
Feature: Reservation Notification Mask

Attribute Name                          Value(Current)
Mask Registration Preeempted Notification    Enabled
Mask Reservation Released Notification       Disabled
Mask Reservation Preempted Notification      Enabled
```

This example shows the reservation persistence feature selected:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x83 Select=0
```

Information similar to the following is displayed:

```
Feature: Reservation Persistance

Attribute Name                          Value(Current)
Persist Through Power Loss                Enabled
```

This example shows `Select=3` (Capabilities) for the arbitration feature:

```
hbacmd nvme-get-feature 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x1 Select=3
```

Information similar to the following is displayed:

```
Feature: Arbitration

Supported Capabilities

Savable:              Yes
Changeable:           Yes
Namespace Specific:   No
```

4.2.2.7 The nvme-list-ns Command

The `nvme-list-ns` command lists the namespaces for a specified NVMe target. This command retrieves two types of namespaces:

- Active namespaces, which are attached to the processor that controls the command
- Allocated namespaces, which have been added to the NVMe subsystem but are not yet connected to a controller

Syntax

`hbacmd nvme-list-ns <WWPN> <Target WWPN> <NQN> [CtrlID=<ctrlid>] [Option]`

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device.
NQN	The qualified name of the NVMe subsystem.
ctrlid	The controller that processes the request (optional), in hexadecimal.
Option	The namespace type (optional); if not used, both active and allocated namespaces are returned: `0` = Get active namespace identifiers `1` = Get allocated namespace identifiers

Support Requirements

- Mandatory for `nvme-list-ns WWPN TWWPN NQN CtrlID 0` (list active namespaces)
- Optional for `nvme-list-ns WWPN TWWPN NQN CtrlID 1` (list allocated namespaces)

Examples

This example shows `option` not specified:

```
hbacmd nvme-list-ns 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz
```

Information similar to the following is displayed:

```
Active Namespaces (attached to controller 0x0001):

Namespace ID : 0x00000001
Namepsece ID : 0x00000002

Allocated Namespaces:

Namespace ID : 0x00000003
Namepsece ID : 0x00000004
```

This example shows active namespace identifiers selected:

```
hbacmd nvme-list-ns 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0
```

Information similar to the following is displayed:

```
Active Namespaces (attached to controller 0x0001):

Namespace ID : 0x00000001
Namepsece ID : 0x00000002
```

This example shows allocated namespace identifiers selected:

```
hbacmd nvme-list-ns 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 1
```

Information similar to the following is displayed:

```
Allocated Namespaces:

Namespace ID : 0x00000003
Namepsece ID : 0x00000004
```

4.2.2.8 The nvme-id-ns Command

The `nvme-id-ns` command sends the NVMe `identify` command for a namespace and displays the results. In addition, if namespace management is supported, the command can retrieve capabilities that are common across all namespaces.

Syntax

`hbacmd nvme-id-ns <WWPN> <Target WWPN> <NQN> <NSID> <Type> [CtrlID=<ctrlid>] [raw]`

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device.
NQN	The qualified name of the NVMe subsystem.
NSID	The namespace identifier, in hexadecimal; use it to list only controllers attached to the specified namespace. Set to `0xFFFFFFFF` to retrieve capabilities that are common across all namespaces.
Type	The namespace identifier type: `0` = Active `1` = Allocated Must be set to `0` if `NDSID=0xFFFFFFFF`.
ctrlid	The controller that processes the request, in hexadecimal.
raw	Specifies that the output is to be provided in hexadecimal format (optional).

NOTE: If the `raw` parameter is not specified, the output displays only a few namespace attributes, in human-readable format. If the `raw` parameter is specified, the entire data structure is displayed in hexadecimal format.

Support Requirements

- Mandatory for `nvme-id-ns WWPN TWWPN NQN NSID 0 [CtrlID=<ctrlid>]` (identify active namespace)
- Optional for `nvme-id-ns WWPN TWWPN NQN NSID 1 [CtrlID=<ctrlid>]` (identify allocated namespace)

Examples

This example identifies active namepaces:

```
hbacmd nvme-id-ns 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x00000001 0
```

Information similar to the following is displayed:

```
Attributes for Namespace 0x00000001:

IEEE Extended Unique Identifier         : 00-00-00-00-00-00-00-00
Size                                    : 32768 (Logical Blocks)
Capacity                                : 32768 (Logical Blocks)
Usage                                   : 32768 (Logical Blocks)
```

```
Logical Block Size                      : 512 Bytes
Multi-path Access                       : Yes
LBA Format                              : 0
Number of Supported LBA Formats         : 0
End-to-end Data Protection              : Disabled
Reservations                            : Not Supported

Namespace Features:
 Thin Provisioning            : Not Supported
 Unwritten Logical Block Error: Not Supported
 NAWUN, NAWUPF, NAWCWU        : Not Supported

End-to-end Protection Capabilities:
 Protection Information Type 1: Supported
 Protection Information Type 2: Supported
 Protection Information Type 3: Supported
```

This example identifies allocated namespaces:

```
hbacmd nvme-id-ns 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x00000001 1
```

Information similar to the following is displayed:

```
Attributes for Namespace 0x00000001:

IEEE Extended Unique Identifier         : 00-00-03-25-01-FF-00-03
Size                                    : 32768 (Logical Blocks)
Capacity                                : 32768 (Logical Blocks)
Usage                                   : 32768 (Logical Blocks)
Logical Block Size                      : 1024 Bytes
Multi-path Access                       : Yes
LBA Format                              : 0
Number of Supported LBA Formats         : 0
End-to-end Data Protection              : Disabled
Reservations                            : Not Supported

Namespace Features:
 Thin Provisioning            : Supported
 Unwritten Logical Block Error: Supported
 NAWUN, NAWUPF, NAWCWU        : Not Supported

End-to-end Protection Capabilities:
 Protection Information Type 1: Supported
 Protection Information Type 2: Not Supported
 Protection Information Type 3: Supported
```

This example shows capabilities that are common across all namespaces:

```
hbacmd nvme-id-ns10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38nqn.2014-
08.com.example:nvme.host.sys.xyz 0xffffffff 0
```

Information similar to the following is displayed:

```
Capabilities common across all namespaces
Metadata Capabilities
   Extended LBA Metadata Transfer      : Supported
   Separate Buffer Metadata Transfer   : Not Supported
```

```
End-To-End Data Protection Capabilities
   First Eight Bytes                    : Supported
   Last Eight Bytes                     : Not Supported

Namespace Multi-path I/O and Sharing Capabilities
   Namespace Sharing                    : Supported

Reservation Capabilities
   Persist Through Power Lost           : Not Supported
   Write Exclusive                      : Not Supported
   Exclusive Access                     : Not Supported
   Write Exclusive - Registrants Only   : Not Supported
   Exclusive Access - Registrants Only : Not Supported
   Write Exclusive - All Registrants    : Not Supported
   Exclusive Access - All Registrants   : Not Supported
   Ignore Existing Key                  : Not Supported

Supported LBA Formats
   LBA Format 0
   LBA Format 1
   LBA Format 2
   LBA Format 3
```

If `NSID` is set to `0xFFFFFFFF`, and namespace management is not supported, the following error message is displayed:

```
ERROR: <804>: Bad NVMe Namespace ID
```

This example shows the `raw` parameter specified:

```
hbacmd nvme-id-ns 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x00000001 1 raw
```

Information similar to the following is displayed:

```
Raw data for namespace 0x00000001:

Standard data bytes:

Byte Dump
0000: 03 04 07 00 00 00 00 60 40 0c 70 06 8c 0a 00 00
0010: 04 02 00 0a 45 4d 55 4c 45 58 20 20 20 20 20 20
0020: 20 20 20 20 00 00 17 6a 41 46 42 52 2d 35 37 46
0030: 35 4d 5a 2d 45 4c 58 20 20 20 20 20 52 03 00 48
0040: 00 3a 00 00 41 41 31 32 32 31 4a 30 43 32 50 20
0050: 20 20 20 20 31 32 30 35 32 38 20 20 68 fa 05 3a
0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
……: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
……: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
……: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
Vendor-specific data bytes:

Byte Dump
0180: 03 04 07 00 00 00 00 60 40 0c 70 06 8c 0a 00 00
0190: 04 02 00 0a 45 4d 55 4c 45 58 20 20 20 20 20 20
01A0: 20 20 20 20 00 00 17 6a 41 46 42 52 2d 35 37 46
01B0: 35 4d 5a 2d 45 4c 58 20 20 20 20 20 52 03 00 48
01C0: 00 3a 00 00 41 41 31 32 32 31 4a 30 43 32 50 20
01D0: 20 20 20 20 31 32 30 35 32 38 20 20 68 fa 05 3a
01E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0FF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

4.2.2.9 The nvme-create-ns Command

The `nvme-create-ns` command creates namespaces on an NVMe subsystem. The command issues the NVMe `namespace` management command to create the namespace and return the results.

Syntax

`hbacmd nvme-create-ns <`*`WWPN`*`> <`*`Target WWPN`*`> <`*`NQN`*`> [CtrlID=<`*`ctrlid`*`>] [`*`Size`*`=<`*`size`*`>] [`*`Capacity`*`=<`*`capacity`*`>] [LBAF=<`*`lbaf`*`>] [MDT=<`*`mdt`*`>][DPS=<`*`dps`*`>] [NMIC=<`*`nmic`*`>]`

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device.
NQN	The qualified name of the NVMe subsystem.
ctrlid	The controller that processes the request, in hexadecimal.
size	The namespace size in logical blocks, in decimal (optional).
capacity	The namespace capacity, in decimal. This value is the maximum number of logical blocks that you can allocate.
lbaf	The LBA format to be used for the namespace (optional).
mdt	The parameter that selects the metadata transfer mechanism (optional): `0` = Transfer metadata as a separate contiguous buffer of data `1` = Transfer metadata at the end of data LBA
dps	The end-to-end data protection type settings, in hexadecimal (optional): `0x0` = Protection information is not enabled. `0x1` = Protection information is enabled and transferred as the last 8 bytes of metadata, Type 1. `0x2` = Protection information is enabled and transferred as the last 8 bytes of metadata, Type 2. `0x3` = Protection information is enabled and transferred as the last 8 bytes of metadata, Type 3. `0x9` = Protection information is enabled and transferred as the first 8 bytes of metadata, Type 1. `0xA` = Protection information is enabled and transferred as the first 8 bytes of metadata, Type 2. `0xB` = Protection information is enabled and transferred as the first 8 bytes of metadata, Type 3.
nmic	The namespace multipath and sharing capabilities (optional): `0` = Multipath disabled. Namespace is private to one controller. `1` = Multipath enabled. Two or more controllers might have access to the namespace.

Support Requirements

- Mandatory if the target supports namespace management.
- Not supported if namespace management is not supported.

Examples

This example shows no optional parameters specified:

```
hbacmd nvme-create-ns 10:00:00:90:fa:e0:63:48 20:03:00:11:0d:a5:70:00 nqn.2014-08.org.nvmexpress:nvm-subsystem-sanblaze
```

The following message is displayed:

```
Successfully created namespace 0x00000003
```

This example shows optional parameters specified:

```
hbacmd nvme-create-ns 10:00:00:90:fa:e0:63:48 20:03:00:11:0d:a5:70:00 nqn.2014-08.org.nvmexpress:nvm-subsystem-sanblaze CtrlID=0x07 Size=32768 Capacity=32768 LBAF=1 MDT=1 DPS=0x9 NMIC=1
```

The following message is displayed:

```
Successfully created namespace 0x00000003
```

This example shows a generic failure:

```
hbacmd nvme-create-ns 10:00:00:90:fa:e0:63:48 20:03:00:11:0d:a5:70:00 nqn.2014-08.org.nvmexpress:nvm-subsystem-sanblaze CtrlID=0x07 Size=32768 Capacity=32768 LBAF=1 MDT=1 DPS=0x1 NMIC=1
```

Messages similar to the following are displayed:

```
ERROR: Failed to create namespace

ERROR: <error code>: <specific error message associated with the returned error code>
```

This example shows an unsupported value for `DPS`, in a case in which namespaces do not support the transfer of data protection information as the last 8 bytes of metadata (`DPS=0x1`):

```
hbacmd nvme-create-ns  10:00:00:90:fa:e0:63:48 20:03:00:11:0d:a5:70:00 nqn.2014-08.org.nvmexpress:nvm-subsystem-sanblaze CtrlID=0x07 Size=32768 Capacity=32768 LBAF=1 MDT=1 DPS=0x1 NMIC=1
```

The following message is displayed:

```
ERROR: Failed to create namespace. This may be due to one of the following conditions:
   -specifying an invalid or unsupported LBA Format
   -enabling an unsupported end-to-end data protection setting

ERROR: <825>: Invalid namespace settings.
```

4.2.2.10 The nvme-delete-ns Command

The `nvme-delete-ns` command deletes namespaces on NVMe subsystems.

Syntax

```
hbacmd nvme-delete-ns <WWPN> <Target WWPN> <NQN> <NSID> [CtrlID=<ctrlid>]
```

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device.

NQN	The qualified name of the NVMe subsystem.
NSID	The namespace identifier to be deleted, in hexadecimal. Set to `all` to delete all namespaces in the subsystem.
ctrlid	The controller that processes the request, in hexadecimal (optional).

Support Requirements

- Mandatory if the target supports namespace management.
- Not supported if namespace management is not supported.

Example

This example deletes a specific namespace:

```
hbacmd nvme-delete-ns 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz 0x00000001
```

This example deletes all namespaces:

```
hbacmd nvme-delete-ns 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz all
```

4.2.2.11 The nvme-attach-ns Command

The `nvme-attach-ns` command attaches a namespace to one or more controllers,

Syntax

`hbacmd nvme-attach-ns <`*WWPN*`> <`*Target WWPN*`> <`*NQN*`> <`*NSID*`> <CtrlID=`*ctrlid1*`[,`*ctrlid2*`,`*ctrlidn*`]>`

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device.
NQN	The qualified name of the NVMe subsystem.
NSID	The namespace identifier to be attached, in hexadecimal.
ctrlid1 ... *ctrlidn*	One or more controller identifiers to which the namespace is to be attached, in hexadecimal. If you are specifying multiple controllers, use commas to delineate them.

Support Requirements

- Mandatory if the target supports namespace management.
- Not supported if namespace management is not supported.

Example

```
hbacmd nvme-attach-ns 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz  0x00000001 ctrlid=0x0001,0x0002
```

4.2.2.12 The nvme-detach-ns Command

The `nvme-detach-ns` command detaches a namespace from one or more controllers to which it is attached.

NOTE: Use this command to detach a namespace from its controllers before using the `nvme-delete-ns` command to delete the namespace.

Syntax

```
hbacmd nvme-detach-ns <WWPN> <Target WWPN> <NQN> <NSID> <CtrlID=ctrlid1[,ctrlid2,ctrlidn]>
```

Parameters

WWPN	The WWPN of the FC initiator port.
Target WWPN	The WWPN of an FC port on an NVMe-capable device.
NQN	The qualified name of the NVMe subsystem.
NSID	The namespace identifier to be detached, in hexadecimal.
ctrlid1 ... ctrlidn	One or more controller identifiers from which the namespace is to be detached, in hexadecimal. If you are specifying multiple controllers, use commas to delineate them.

Support Requirements

- Mandatory if the target supports namespace management.
- Not supported if namespace management is not supported.

Example

```
hbacmd nvme-detach-ns 10:00:00:90:fa:5d:05:a9 50:06:01:60:90:20:5C:38 nqn.2014-
08.com.example:nvme.host.sys.xyz A3-45-F7-A7 0x00000001 ctrlid=0x0001,0x0002
```

Appendix A: Configuring NVMe over FC on a Target (SLES 12 SP3 Only)

NOTE: Before configuring NVMe over FC using native NVMe CLI commands, ensure that you have installed the latest Emulex firmware for the LPe31000-series, LPe32000-series, and LPe35000-series adapters.

NOTE: At this time, NVME over FC on target devices is supported on SLES 12 SP3 only.

A.1 Configuring the lpfc.conf File

To create and configure the `lpfc.conf` file, perform the following steps:

1. Find the WWPNs of HBAs that are available to be used as NVMe target ports by typing the following command:
   ```
   cat /sys/class/fc_host/host*/port_name
   ```
   Information similar to the following is displayed (the low number is usually port 0 of the adapter):
   ```
   0x10000090fa931110
   0x10000090fa931111
   ```
   Record the WWPNs of the LPe31000-series, LPe32000-series, and LPe35000-series adapters you want to use as target ports; you will need this information in Step 5.

2. Find the WWNNs of the HBAs in Step 1 by typing the following command:
   ```
   cat /sys/class/fc_host/host*/node_name
   ```
   Information similar to the following is displayed:
   ```
   0x20000090fa931110
   0x20000090fa931111
   ```
   Record the WWNNs of the LPe31000-series, LPe32000-series, and LPe35000-series adapters you want to use as target ports; you will need this information in Step 5.

3. NVMe subsystems and namespaces are created from the storage block devices on the target. An existing storage block device on the target can be used by NVMe namespaces as their storage media. Display the available storage block devices to share over the NVMe fabric by typing the following command:
   ```
   lsblk
   ```
   All block devices found by the target are displayed, for example:

```
 NAME                                                MAJ:MIN  RM   SIZE      RO   TYPE     MOUNTPOINT
sda                                                    8:0    0   136.7G    0    disk
├─sda1                                                 8:1    0   2G        0    part
├─sda2                                                 8:2    0   40G       0    part
├─sda3                                                 8:3    0   94.7G     0    part
└─3600508b1001037383941424344450400                  254:0    0   136.7G    0    mpath
  ├─3600508b1001037383941424344450400-part1          254:1    0   2G        0    part     [SWAP]
  ├─3600508b1001037383941424344450400-part2          254:2    0   40G       0    part     /var/opt
  └─3600508b1001037383941424344450400-part3          254:3    0   94.7G     0    part     /home
sdb                                                    8:16   0   3G        0    disk
  └─3600000e00d2a0000002a017300340000                  254:4    0   3G        0    mpath
sdc                                                    8:32   0   3G        0    disk
```

```
  └─3600000e00d2a0000002a017300350000          254:5   0     3G        0     mpath
sdd                                             8:48   0     3G        0     disk
  └─3600000e00d2a0000002a017300360000          254:6   0     3G        0     mpath
sde                                             8:64   0     3G        0     disk
  └─3600000e00d2a0000002a017300370000          254:7   0     3G        0     mpath
sdf                                             8:80   0     3G        0     disk
  └─3600000e00d2a0000002a017300380000          254:8   0     3G        0     mpath
```

Record the block devices you want to use, for example `/dev/sde` and `/dev/sdf`.

NOTE: Block devices are located in the `/dev` directory.

4. Navigate to the `/etc/modprobe.d` subdirectory, and create a file with the driver name `lpfc.conf`.
5. Insert the following line in the `lpfc.conf` file:
   `options lpfc lpfc_enable_nvmet=<WWPN1, WWPN2> lpfc_enable_fc4_type=3`
   where <*WWPN1, WWPN2*> are the WWPNs from Step 1.

NOTE: `lpfc_enable_fc4_type=3` allows both SCSI and NVMe over FC protocols to be enabled on the port; by default, the port is an initiator for the SCSI protocol. The WWPNs that are listed indicate FC ports that are exclusively NVMe over FC targets; the SCSI protocol is not used on these ports.

6. Save the `lpfc.conf` file.
7. Regenerate the ramdisk for the server by typing the following command:
   `#dracut --force`
8. Reboot the system.

   After the system is rebooted, the configured protocols are enacted on the FC ports, and the ports identified by the WWPNs in Step 5 are now running as target ports.

A.2 Configuring Subsystems, Namespaces, and Ports

Use the NVMe CLI for target systems to create the target subsystems, namespaces, and ports, and save them to a `.json` file, by performing the following procedure:

1. Start the NVMe CLI for target systems by typing the following command:
   `nvmetcli`

NOTE: A warning message might appear the first time you run this command. Ignore the message.

A separate utility screen appears. Step 2 though Step 13 are carried out in this screen.

2. Navigate to the `/subsystems` directory, and create a subsystem, such as `fctest1`, by typing the following command:
   `create fctest1`
3. Navigate to the `/subsystems/fctest1` subdirectory, and set the `allow_any_host` parameter to `1`, by typing the following command:
   `set attr allow_any_host=1`
4. Navigate to the `/subsystems/fctest1/namespaces` subdirectory, and create `namespace 1` by typing the following command:
   `create 1`

5. Navigate to the `/subsystems/fctest1/namespaces/1` subdirectory, and set the device path to the backing storage device you recorded previously in Step 3 in Section A.1, Configuring the lpfc.conf File, by typing the following command:
   `set device path=/dev/<blockdevice>`
   For example:
   `set device path=/dev/sde`
   Information similar to the following is displayed:
   `Parameter path is now '/dev/sde'.`

NOTE: This step causes the namespace to use the specified backing storage device, which already exists, as its storage media.

6. Enable namespace 1 by typing the following command:
   `enable`
   Information similar to the following is displayed:
   `The Namespace has been enabled.`
   The target NVMe over FC subsystems and namespaces have been created. You can create and enable additional namespaces by repeating Step 4 through Step 6 in this section, incrementing the namespace number each time.
7. After you have created all of the namespaces, navigate to the `/ports` directory and create a link to the desired port; in this case, Port 1, by typing the following command:
   `create 1`
8. Navigate to the `/ports/1` subdirectory, and set the NVMe type (the `trtype` and `adrfram` parameters) to `fc`:
   a. Type the following command:
      `set addr trtype=fc`
      Information similar to the following is displayed:
      `Parameter trtype is now 'fc'.`
   b. Type the following command:
      `set addr adrfam=fc`
      Information similar to the following is displayed:
      `Parameter adrfam is now 'fc'.`
9. Set the network address (the `traddr` parameter) to the WWNN and WWPN of the target HBA obtained in Step 1 and Step 2 in Section A.1, Configuring the lpfc.conf File by typing the following command:
   `set addr traddr=nn-<WWNN1>:pn-<WWPN1>`
   For example:
   `set addr traddr=nn-0x20000090fa931110:pn-0x10000090fa931110`
   Information similar to the following is displayed:
   `Parameter traddr is now 'nn-0x20000090fa931110:pn-0x10000090fa931110'`
10. Set the transport service ID (the `trsvcid` parameter) to `none` by typing the following command:
    `set addr trsvcid=none`
    Information similar to the following is displayed:
    `Parameter trsvcid is now 'none'.`
    The port connection has been created. You can create additional port connections by repeating Step 7 through Step 10 in this section, incrementing the port number each time.
11. Link the port to the subsystem by navigating to the `/ports/1/subsystems` directory and typing the following command:
    `create <subsystem>`
    where <subsystem> is the same name as the subsystem you created in Step 2 in this section.

For example:
`create fctest1`

12. Save the `.json` file by typing the following command:
`saveconfig <filename>.json`

NOTE: If you want the configuration to be restored automatically after rebooting, this file must be named `config.json`, and it must be saved under the path `/etc/nvmet/`. Otherwise, <*filename*> can be any valid file name.

13. Exit the NVMe CLI for target systems utility screen by typing the following command:
`exit`
The target configuration is now stored in the <*filename*>`.json` file in the directory that was current before you started the NVMeT CLI, or in the `/etc/nvmet/` directory.

Unless you want to change the subsystems, namespaces, or port configurations, you perform Step 1 through Step 13 in this section only the first time you set the configuration.

After configuring the target, you can run the `nvme_info` command to verify that your configuration is working by typing the following command on the target:
`cat /sys/class/scsi_host/host<X>/nvme_info`

where <*X*> is the host number.

Information similar to the following appears:

```
NVME Target Enabled  State REGISTERED
NVME Target: lpfc0 WWPN x10000090fa942779 WWNN x20000090fa942779 DID x011400

NVME Target: Statistics
LS: Rcv 00000004 Drop 00000000 Abort 00000000
LS: Xmt 00000004 Drop 00000000 Cmpl 00000004 Err 00000000
FCP: Rcv 00000002 Defer 00000000 Release 00000002 Drop 00000000
FCP Rsp: RD 00000000 rsp 00000000 WR 00000002 rsp 00000002 drop 00000000
FCP Rsp Cmpl: 00000004 err 00000000 drop 00000000
ABORT: Xmt 00000000 Cmpl 00000000
ABORT: Sol 00000000  Usol 00000000 Err 00000000 Cmpl 00000000
IO_CTX: 00001706  WAIT: cur 00000000 tot 00000000
CTX Outstanding 00000000
```

The NVMe over FC target configuration is now completed. You must now configure NVMe over FC on an initiator.

If you reboot the system, or if you need to restore the NVMe target, you can type the following command:
`nvmetcli restore <filename>.conf`

NOTE: If a situation arises in which the initiator system boots before the target system, you must use the Linux `issue_lip` command after the target system is up. This command allows the initiator to discover the targets. Type the following command:

```
echo 1 > /sys/class/fc_host/host<X>/issue_lip
```

where `host`<*X*> represents the host number.

If you want to clear the active configuration, perform the following steps:

1. Disable the initiator links.
2. Reboot the initiator.
3. Clear the active configuration by typing the following command:
`nvmetcli clear`

4. Reconfigure the target by following the instructions in Step 1 through Step 13 in this section.
5. Enable the initiator links.

You can set the saved configuration to be restored automatically after a reboot by using `systemctl` to enable the `nvmet.service`. Type the following command:

```
systemctl enable nvmet.service
```

NOTE: The service requires that the `.json` file must be named `config.json`, and it must be saved under the path `/etc/nvmet/`, for it to be loaded automatically at boot time.

To disable the `nvmet.service` using `systemctl`, type the following command:

```
systemctl disable nvmet.service
```

Appendix B: Multipathing

SLES 15 includes support for native NVMe multipathing, This support changes the manner in which NVMe devices are presented, especially in cases where multiple paths to target namespaces exist.

The additional detection and management of multipath NVME configurations has changed the manner in which NVMe devices are presented and related in `/sys/class/nvme`. These changes significantly affect the views generated by the kernel and system utilities, such as `nvme-cli`. This appendix describes these new views.

In SLES 12 SP3, if an NVMe device is connected to a subsystem, a controller element is created. If the subsystem is connected using multiple paths, multiple controller elements are created. Each controller element is considered a unique and separate storage entity, even though the subsystem and namespaces the controller could access might be the same.

In SLES 15, controller elements are still created for each connection to a subsystem. However, when a controller is created, the subsystem is compared to the list of subsystems that have already been found in the system:

- If the subsystem is not found, a new subsystem element is created and added to the system list, with the controller linked to it as a path.
- If the subsystem is found, the controller is linked to the subsystem as an additional path.

Namespaces are elements of the NVMe subsystem, and not the NVMe controller. A controller acts as a communication port to a subsystem. If a subsystem contains two controllers (communication ports), and namespace 1 is changed by controller A, controller B sees this change as an action on namespace 1, because the namespace is a singular component of the subsystem.

In SLES 12 SP3, because each controller is seen as unique, namespaces for the subsystem are created for every controller the namespace is found on. Processing of I/O requests by the device name are handled only by the controller, and they are blindly passed on to the storage device. If concurrent actions occur on namespaces on different controllers connected to the same subsystem, the application or administrator must realize that the namespace is a single entity behind the controllers, and the application or administrator must access the two controllers accordingly, so as to provide data coherency and to avoid data corruption.

In SLES 15, namespace device names are created when the device initially connects to the subsystem, and the controller is the first path to the subsystem. The device name reflects the subsystem that the controller is part of. If multiple controllers connect to the same subsystem, no additional namespace device names are created, because the controllers are simply paths that provide access to the same storage device. To maintain compatibility with utilities that are operating on NVMe devices on SLES 12 SP3, the name format used for the namespace device name has been kept the same. This can present some confusion, because the namespace device name (for example, `/dev/nvme0n1`) contains an `nvme<x>` prefix that, on SLES 12 SP3, was the controller name, but on SLES 15 is a prefix for the subsystem instance. In addition, for compatibility with management of NVMe controllers, the device name used for NVMe controllers (for example, `/dev/nvme2`) was left the same in SLES 15. So, in SLES 15, you might see both a `/dev/nvme3` controller name and a `/dev/nvme3n1` namespace name, but no correlation exists between the *nvme3* portion of both names. In summary: In SLES 15, the name `/dev/nvme<A>n<x>` means *namespace number* <x> *on subsystem instance* <A>, whereas the name `/dev/nvme<A>` means *controller instance* <A>. The only way the two names correlate is if controller instance <*A*> actually connects to subsystem instance <*A*>, and is therefore a path to subsystem instance <*A*>

Beyond the device name difference, the other place this change in behavior is seen is in system utilities, such as `nvme-cli`. For example, the `nvme list` command lists all NVMe namespace devices.

NOTE: This document covers are two similar commands:

- `nvme list` (with a space) is a native NVMe CLI command.
- `nvme-list` (with a hyphen) is an Emulex OneCommand Manager CLI command.

This chapter refers to the `nvme list` command.

Thus, if two controllers are connected to a single subsystem with a single namespace, information similar to the following is displayed in SLES 12 SP3:

```
Node          SN                   Model   Namespace   Usage                  Format         FW Rev
/dev/nvme0n1  0000000000000000     Linux   1           268.44 GB / 268.44 GB  512 B +  0 B   4.4.131-
/dev/nvme1n1  0000000000000000     Linux   1           268.44 GB / 268.44 GB  512 B +  0 B   4.4.131-
```

On the other hand, information similar to the following is displayed in SLES 15:

```
Node          SN                   Model   Namespace   Usage                  Format         FW Rev
/dev/nvme0n1  0000000000000000     Linux   1           268.44 GB / 268.44 GB  512 B +  0 B   4.4.131-
```

In SLES 15, you can display the list of subsystems that have been detected by the system using the new `nvme list-subsys` command. For example:

```
nvme list-subsys
nvme-subsys0 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:19d48a6d-3644-4f65-8046-80bd88fe45a6
\
 +- nvme0 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f1f:pn-
0x100000109b346f1f
 +- nvme3 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f20:pn-
0x100000109b346f20
nvme-subsys1 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:2c6d862f-b77e-43b1-b345-7f990125088e
\
 +- nvme1 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f1f:pn-
0x100000109b346f1f
 +- nvme6 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f20:pn-
0x100000109b346f20
nvme-subsys2 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:4bbbd877-6de2-4a67-ad56-e0333e2437b5
\
 +- nvme2 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f1f:pn-
0x100000109b346f1f
 +- nvme7 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f20:pn-
0x100000109b346f20
nvme-subsys3 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:c8da7263-fefe-4563-a14c-a9f0362dcb03
\
 +- nvme5 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f1f:pn-
0x100000109b346f1f
 +- nvme9 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f20:pn-
0x100000109b346f20
nvme-subsys4 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:5f1d7788-fab1-45cf-af9c-d30dc9a9ce89
\
 +- nvme4 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f1f:pn-
0x100000109b346f1f
 +- nvme8 fc traddr=nn-0x20000090fa942e16:pn-0x10000090fa942e16 host_traddr=nn-0x200000109b346f20:pn-
0x100000109b346f20
```

In the above example, five subsystems are displayed. Each subsystem contains multiple controllers representing multiple paths. Note that the `nvme-subsys` numbering is disassociated from the component controller identification, as demonstrated by `nvme-subsys3`, and controllers `nvme5` and `nvme9`.

In the preceding example configuration, each of the five subsystems contain four namespaces. The `nvme list-subsys` view shows two controllers. In SLES 12 SP3, each namespace and path is uniquely presented to the operating system. In SLES 15, each namespace is displayed once.

```
nvme list
Node           SN                   Model   Namespace   Usage                    Format         FW Rev
/dev/nvme0n1   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme0n2   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme0n3   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme0n4   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme1n1   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme1n2   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme1n3   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme1n4   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme2n1   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme2n2   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme2n3   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme2n4   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme3n1   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme3n2   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme3n3   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme3n4   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme4n1   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme4n2   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme4n3   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme4n4   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme5n1   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme5n2   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme5n3   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
/dev/nvme5n4   0000000000000000     Linux   1           268.44 GB / 268.44 GB    512 B +  0 B   4.4.131-
```

Note that the naming includes the subsystem and namespace numbering in the form `/dev/nvme{`*`subsystem instance`*`}n{`*`namespace`*`}`. The controller (or path) is not shown.

You can also view the device nodes using the `ls` command:

```
ls -l  /dev/nvme*
0 crw------- 1 root root  10,  57 Jun 10 16:02 /dev/nvme-fabrics
0 crw------- 1 root root 246,   0 Jun 10 16:02 /dev/nvme0
0 brw-rw---- 1 root disk 259, 231 Jun 10 16:02 /dev/nvme0n1
0 brw-rw---- 1 root disk 259, 271 Jun 10 16:02 /dev/nvme0n2
0 brw-rw---- 1 root disk 259, 303 Jun 10 16:02 /dev/nvme0n3
0 brw-rw---- 1 root disk 259, 311 Jun 10 16:02 /dev/nvme0n4
0 crw------- 1 root root 246,   1 Jun 10 16:02 /dev/nvme1
0 brw-rw---- 1 root disk 259, 323 Jun 10 16:02 /dev/nvme1n1
```

```
0 brw-rw---- 1 root disk 259, 343 Jun 10 16:02 /dev/nvme1n2
0 brw-rw---- 1 root disk 259, 347 Jun 10 16:02 /dev/nvme1n3
0 brw-rw---- 1 root disk 259, 351 Jun 10 16:02 /dev/nvme1n4
0 crw------- 1 root root 246,   2 Jun 10 16:02 /dev/nvme2
0 brw-rw---- 1 root disk 259, 359 Jun 10 16:02 /dev/nvme2n1
0 brw-rw---- 1 root disk 259, 363 Jun 10 16:02 /dev/nvme2n2
0 brw-rw---- 1 root disk 259, 381 Jun 10 16:02 /dev/nvme2n3
0 brw-rw---- 1 root disk 259, 383 Jun 10 16:02 /dev/nvme2n4
0 crw------- 1 root root 246,   3 Jun 10 16:02 /dev/nvme3
0 brw-rw---- 1 root disk 259, 397 Jun 10 16:02 /dev/nvme3n1
0 brw-rw---- 1 root disk 259, 399 Jun 10 16:02 /dev/nvme3n2
0 brw-rw---- 1 root disk 259, 401 Jun 10 16:02 /dev/nvme3n3
0 brw-rw---- 1 root disk 259, 403 Jun 10 16:02 /dev/nvme3n4
0 crw------- 1 root root 246,   4 Jun 10 16:02 /dev/nvme4
0 brw-rw---- 1 root disk 259, 385 Jun 10 16:02 /dev/nvme4n1
0 brw-rw---- 1 root disk 259, 387 Jun 10 16:02 /dev/nvme4n2
0 brw-rw---- 1 root disk 259, 389 Jun 10 16:02 /dev/nvme4n3
0 brw-rw---- 1 root disk 259, 391 Jun 10 16:02 /dev/nvme4n4
0 crw------- 1 root root 246,   5 Jun 10 16:02 /dev/nvme5
0 crw------- 1 root root 246,   6 Jun 10 16:02 /dev/nvme6
0 crw------- 1 root root 246,   7 Jun 10 16:02 /dev/nvme7
0 crw------- 1 root root 246,   8 Jun 10 16:02 /dev/nvme8
0 crw------- 1 root root 246,   9 Jun 10 16:02 /dev/nvme9
```

This example demonstrates that one device node is created per namespace, with the form `/dev/nvme{`*`subsystem instance`*`}n{`*`namespace`*`}`.

Additionally, nodes are created for the controller instances `/dev/nvme{`*`controller instance`*`}`, but it is important to note that the subsystem instance and the controller instance are not correlated, as demonstrated by the previous `nvme list-subsys` example.

You can also list all block devices that are NVMe devices using the `ls` command. The file names are displayed as symbolic links to the actual block devices.

```
ls -l /sys/block/nvme*
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0c222n1 -> ../devices/virtual/nvme-fabrics/ctl/
nvme0/nvme0c222n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0c222n2 -> ../devices/virtual/nvme-fabrics/ctl/
nvme0/nvme0c222n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0c222n3 -> ../devices/virtual/nvme-fabrics/ctl/
nvme0/nvme0c222n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0c222n4 -> ../devices/virtual/nvme-fabrics/ctl/
nvme0/nvme0c222n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0c227n1 -> ../devices/virtual/nvme-fabrics/ctl/
nvme3/nvme0c227n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0c227n2 -> ../devices/virtual/nvme-fabrics/ctl/
nvme3/nvme0c227n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0c227n3 -> ../devices/virtual/nvme-fabrics/ctl/
nvme3/nvme0c227n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0c227n4 -> ../devices/virtual/nvme-fabrics/ctl/
nvme3/nvme0c227n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0n1 -> ../devices/virtual/nvme-subsystem/nvme-
subsys0/nvme0n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0n2 -> ../devices/virtual/nvme-subsystem/nvme-
subsys0/nvme0n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0n3 -> ../devices/virtual/nvme-subsystem/nvme-
subsys0/nvme0n3
```

```
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme0n4 -> ../devices/virtual/nvme-subsystem/nvme-
subsys0/nvme0n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1c223n1 -> ../devices/virtual/nvme-fabrics/ctl/
nvme1/nvme1c223n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1c223n2 -> ../devices/virtual/nvme-fabrics/ctl/
nvme1/nvme1c223n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1c223n3 -> ../devices/virtual/nvme-fabrics/ctl/
nvme1/nvme1c223n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1c223n4 -> ../devices/virtual/nvme-fabrics/ctl/
nvme1/nvme1c223n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1c229n1 -> ../devices/virtual/nvme-fabrics/ctl/
nvme6/nvme1c229n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1c229n2 -> ../devices/virtual/nvme-fabrics/ctl/
nvme6/nvme1c229n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1c229n3 -> ../devices/virtual/nvme-fabrics/ctl/
nvme6/nvme1c229n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1c229n4 -> ../devices/virtual/nvme-fabrics/ctl/
nvme6/nvme1c229n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1n1 -> ../devices/virtual/nvme-subsystem/nvme-
subsys1/nvme1n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1n2 -> ../devices/virtual/nvme-subsystem/nvme-
subsys1/nvme1n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1n3 -> ../devices/virtual/nvme-subsystem/nvme-
subsys1/nvme1n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme1n4 -> ../devices/virtual/nvme-subsystem/nvme-
subsys1/nvme1n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2c224n1 -> ../devices/virtual/nvme-fabrics/ctl/
nvme2/nvme2c224n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2c224n2 -> ../devices/virtual/nvme-fabrics/ctl/
nvme2/nvme2c224n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2c224n3 -> ../devices/virtual/nvme-fabrics/ctl/
nvme2/nvme2c224n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2c224n4 -> ../devices/virtual/nvme-fabrics/ctl/
nvme2/nvme2c224n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2c230n1 -> ../devices/virtual/nvme-fabrics/ctl/
nvme7/nvme2c230n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2c230n2 -> ../devices/virtual/nvme-fabrics/ctl/
nvme7/nvme2c230n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2c230n3 -> ../devices/virtual/nvme-fabrics/ctl/
nvme7/nvme2c230n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2c230n4 -> ../devices/virtual/nvme-fabrics/ctl/
nvme7/nvme2c230n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2n1 -> ../devices/virtual/nvme-subsystem/nvme-
subsys2/nvme2n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2n2 -> ../devices/virtual/nvme-subsystem/nvme-
subsys2/nvme2n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2n3 -> ../devices/virtual/nvme-subsystem/nvme-
subsys2/nvme2n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme2n4 -> ../devices/virtual/nvme-subsystem/nvme-
subsys2/nvme2n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3c228n1 -> ../devices/virtual/nvme-fabrics/ctl/
nvme5/nvme3c228n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3c228n2 -> ../devices/virtual/nvme-fabrics/ctl/
nvme5/nvme3c228n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3c228n3 -> ../devices/virtual/nvme-fabrics/ctl/
nvme5/nvme3c228n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3c228n4 -> ../devices/virtual/nvme-fabrics/ctl/
nvme5/nvme3c228n4
```

```
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3c232n1 -> ../devices/virtual/nvme-fabrics/ctl/
nvme9/nvme3c232n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3c232n2 -> ../devices/virtual/nvme-fabrics/ctl/
nvme9/nvme3c232n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3c232n3 -> ../devices/virtual/nvme-fabrics/ctl/
nvme9/nvme3c232n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3c232n4 -> ../devices/virtual/nvme-fabrics/ctl/
nvme9/nvme3c232n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3n1 -> ../devices/virtual/nvme-subsystem/nvme-
subsys3/nvme3n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3n2 -> ../devices/virtual/nvme-subsystem/nvme-
subsys3/nvme3n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3n3 -> ../devices/virtual/nvme-subsystem/nvme-
subsys3/nvme3n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme3n4 -> ../devices/virtual/nvme-subsystem/nvme-
subsys3/nvme3n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4c226n1 -> ../devices/virtual/nvme-fabrics/ctl/
nvme4/nvme4c226n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4c226n2 -> ../devices/virtual/nvme-fabrics/ctl/
nvme4/nvme4c226n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4c226n3 -> ../devices/virtual/nvme-fabrics/ctl/
nvme4/nvme4c226n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4c226n4 -> ../devices/virtual/nvme-fabrics/ctl/
nvme4/nvme4c226n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4c231n1 -> ../devices/virtual/nvme-fabrics/ctl/
nvme8/nvme4c231n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4c231n2 -> ../devices/virtual/nvme-fabrics/ctl/
nvme8/nvme4c231n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4c231n3 -> ../devices/virtual/nvme-fabrics/ctl/
nvme8/nvme4c231n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4c231n4 -> ../devices/virtual/nvme-fabrics/ctl/
nvme8/nvme4c231n4
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4n1 -> ../devices/virtual/nvme-subsystem/nvme-
subsys4/nvme4n1
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4n2 -> ../devices/virtual/nvme-subsystem/nvme-
subsys4/nvme4n2
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4n3 -> ../devices/virtual/nvme-subsystem/nvme-
subsys4/nvme4n3
lrwxrwxrwx 1 root root 0 Jun 11 09:56 /sys/block/nvme4n4 -> ../devices/virtual/nvme-subsystem/nvme-
subsys4/nvme4n4
```

In the previous example, the symbolic links illustrate that the available namespaces are assigned a virtual controller (or path) number that is uniquely associated with the subsystem.


BROADCOM®