



# NVMesh User Guide

2.5.2 — Last update: 2 November 2022

Excelero, Ltd.

# Table of Contents

- 1. Copyright and Trademark Information ..... 5**
- 2. Preface ..... 6**
- 3. Introduction To NVMeSH Technology ..... 7**
  - 3.1. Overview ..... 7
  - 3.2. Deployment and Nomenclature ..... 9
  - 3.3. Target and Client Node Communication ..... 9
  - 3.4. Software Components ..... 9
    - 3.4.1. Management ..... 10
    - 3.4.2. Target ..... 10
    - 3.4.3. Client ..... 11
      - 3.4.3.1. Multiple Client Instances ..... 12
      - 3.4.3.2. NVMe-over-Fabrics Gateway ..... 13
  - 3.5. High Availability ..... 13
  - 3.6. Load Balancing ..... 14
  - 3.7. Drive Format Types ..... 14
  - 3.8. Logical Volume Types ..... 16
    - 3.8.1. MeshProtect Concatenated ..... 16
    - 3.8.2. MeshProtect 0 (Striped RAID-0) ..... 18
    - 3.8.3. MeshProtect 1 (Mirrored) ..... 20
    - 3.8.4. MeshProtect 10 (Striped & Mirrored) ..... 21
    - 3.8.5. MeshProtect 60 (Dual Parity) ..... 22
      - 3.8.5.1. Target Node Redundancy ..... 24
  - 3.9. Access Modes ..... 24
  - 3.10. Zones ..... 25
  - 3.11. Minimum Configurations ..... 26
  - 3.12. Security ..... 28
    - 3.12.1. Securing Volume Attachments ..... 28
  - 3.13. CRC Check ..... 29
  - 3.14. 512-byte Block Size Emulation ..... 30
- 4. NVMeSH Software Installation ..... 31**
  - 4.1. Installation Overview ..... 31
  - 4.2. Prerequisites ..... 32
    - 4.2.1. Hardware Requirements ..... 32
      - 4.2.1.1. Memory Requirements ..... 32
      - 4.2.1.2. NVMe Device Requirements ..... 33
      - 4.2.1.3. NIC Requirements ..... 34
    - 4.2.2. Software Requirements ..... 34
      - 4.2.2.1. Operating System ..... 36

4.2.2.2. Mellanox OFED .....	36
4.2.3. Network Connectivity .....	37
4.2.4. Firewalls and Specific Ports .....	39
4.2.5. NTP Time Synchronization.....	40
4.3. Network Configuration.....	40
4.3.1. RoCEv2 .....	40
4.3.1.1. Global Pause .....	41
4.3.1.2. PFC/ECN.....	41
4.3.1.3. Lossy RoCE.....	50
4.3.2. RoCEv2 Multipathing .....	50
4.3.2.1. Dual Switch Requirements.....	51
4.3.2.2. Linux OS Requirements.....	52
4.3.2.3. Switch Configuration Examples.....	53
4.3.2.4. Linux OS Configuration Examples .....	57
4.3.2.4.1. ifcfg Files.....	58
4.3.2.4.2. rt_tables File .....	59
4.3.2.4.3. route Files .....	60
4.3.2.4.4. rule Files .....	61
4.3.3. TCP/IP Support.....	62
4.4. Software Delivery .....	63
4.4.1. Software Delivery Red Hat / CentOS.....	63
4.4.2. Software Delivery Ubuntu .....	64
4.5. Installation Instructions.....	65
4.5.1. Install MongoDB and NodeJS .....	65
4.5.2. Install the NVMesh Management Server .....	68
4.5.2.1. Deploying NVMesh Management in Containers .....	69
4.5.3. Install the NVMesh Clients and Targets .....	77
4.5.4. Enable RDDA.....	78
4.5.5. Configure TCP/IP Support.....	80
4.5.6. Start the NVMesh Client.....	80
4.5.7. Exclude Drives.....	81
4.5.8. Start the NVMesh Target .....	82
4.5.9. Add non-NVMe Drives .....	83
4.5.10. (Optional) Assign Targets to Zones.....	83
4.5.11. Format Drives .....	84
<b>5. Getting Started .....</b>	<b>85</b>
5.1. Drives Management .....	85
5.2. Volume Creation and Attachment .....	86
5.2.1. Login to the Management GUI .....	87
5.2.2. Verify Client and Target Registration.....	92
5.2.3. Create a Volume .....	94
5.2.4. Attach a Volume to a Client.....	97

<b>6. General Settings</b>	<b>99</b>
<b>7. Client and Target Configuration</b>	<b>102</b>
7.1. Configuration Profiles	102
7.2. Client and Target Options	103
7.2.1. Client and Target Shared Options	104
7.2.2. Client Specific Options	107
7.2.3. Target Specific Options	109
7.3. Module Parameters	111
<b>8. Storage Configuration</b>	<b>126</b>
8.1. Target Classes	126
8.2. Drive Classes	127
8.3. Volume Provisioning Groups	129
8.4. Protection Domains	132
8.5. Read-only Access	133
8.6. Command Line Client Operations	133
8.6.1. Attaching Volumes	134
8.6.2. Detaching Volumes	136
8.6.3. Checking Volume Status	137
8.7. Drive Segment Zeroing	140
8.8. Volume Rebuild Prioritization	140
<b>9. Management Configuration</b>	<b>142</b>
9.1. Management Server Options	142
9.1.1. General Options	142
9.1.2. SSL Configuration Options	146
9.1.3. MongoDB Connection Options	147
9.1.4. Log Level Options	150
9.1.5. SMTP Options	154
9.1.6. Backup Options	157
9.1.7. Statistics Options	158
9.2. Management Scalability	163
9.2.1. MongoDB Replica-Set	163
9.2.2. Update Management Servers Configuration	165
9.2.3. Using Load Balancers with HA Management	166
<b>10. Monitoring</b>	<b>167</b>
10.1. Dashboard	167
10.2. Volume State	168
10.3. Client State	171
10.4. Target State	171
10.5. Statistics	172
10.5.1. Customized Statistics Dashboard	172

10.5.2. Statistics Gadgets .....	175
10.5.3. Statistics Rollup .....	175
10.6. /proc Statistics.....	175
10.6.1. Client Volume Statistics .....	175
10.6.2. Client Drive Statistics .....	177
10.7. /proc Monitoring .....	179
10.8. About Screen .....	182
10.9. Node Health Check (NHC) .....	184
<b>11. Logging .....</b>	<b>185</b>
11.1. System Logs .....	185
11.2. Management Logs.....	186
11.3. Binary Tracing.....	187
<b>12. Alerts.....</b>	<b>189</b>
<b>13. NVMesh Best Practices .....</b>	<b>192</b>
13.1. Performance Best Practices .....	192
13.1.1. Homogeneous OFED Configuration .....	192
13.1.2. CPU Interrupt Affinity and IRQ Balancing.....	192
13.1.3. Tuned .....	195
13.1.4. NUMA Architectures.....	196
13.2. Multi-Path Configuration .....	197
13.2.1. Ethernet & RoCE .....	198
13.2.2. RDMA Atomics Limitations .....	199
13.3. Hardware Related Items .....	199
13.3.1. NVMe Drives.....	199
13.3.2. PCIe Considerations .....	200
13.3.3. Network Interface Cards .....	202
13.4. Network Considerations .....	202
13.5. Kernel Configuration.....	203
13.6. MongoDB Best Practices.....	204
13.7. Best Practices for AMD EPYC Processors .....	205
<b>14. Maintenance.....</b>	<b>208</b>
14.1. Support .....	208
14.2. Troubleshooting .....	208
14.2.1. nvmesh_health_check.....	208
14.2.2. nvmesh_logs_collector.....	212
14.3. Hardware Operations .....	213
14.3.1. Drive Failure & Replacement.....	213
14.3.2. NIC Failure and Replacement .....	215
14.3.3. Moving a Drive.....	216
14.3.4. Drive Resize .....	216

14.4. Software Operations.....	216
14.4.1. Start NVMesh.....	217
14.4.2. Shutdown NVMesh .....	217
14.4.3. Uninstall the NVMesh Software .....	218
14.4.4. Modify Management Server IP .....	218
14.4.5. Reset Factory Defaults.....	218
14.4.6. Rename Hostname .....	219
14.4.7. Upgrade NVMesh.....	219
14.4.7.1. Upgrading from NVMesh 1.3 or older .....	220
14.4.7.2. Upgrading from NVMesh 2.0 or newer .....	220
14.4.7.3. Upgrading NVMesh Management in Docker Container.....	222
14.4.8. Advanced Operations .....	223
14.4.8.1. Disabling Rebuild Operations.....	223
<b>15. Configuration Limits.....</b>	<b>225</b>
15.1. Client Limitations.....	225
15.2. Drive Limitations .....	225
15.3. Drive Class Limitations .....	225
15.4. Key Pair Limitations .....	225
15.5. Networking Limitations .....	226
15.6. Target Limitations.....	226
15.7. Target Class Limitations .....	226
15.8. User Limitations .....	226
15.9. Volume Limitations .....	227
15.10. Volume Provisioning Group Limitations .....	227
15.11. Volume Security Group Limitations.....	228
<b>16. Reference Information.....</b>	<b>229</b>
16.1. SystemD Reference Table.....	229
<b>17. Document Reference .....</b>	<b>231</b>

# 1. Copyright and Trademark Information

---

© 2015-2022 Excelero, Inc. All rights reserved. Specifications are subject to change without notice. Excelero, the Excelero logo, MeshProtect, and Remote-Direct-Drive-Access (RDDA) are trademarks of Excelero, Inc. in the United States and/or other countries. NVMesh® is a registered trademark of Excelero, Inc. in the United States.

Mellanox and ConnectX are registered trademarks of NVIDIA.

Intel is a registered trademark of Intel Corporation. Xeon and Core are trademarks of Intel Corporation. All other brands or products are trademarks or registered trademarks of their respective holders and should be treated as such.

Red Hat, Red Hat Enterprise Linux, the Red Hat logo and OpenShift are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

Ubuntu® is a trademark of Canonical or its subsidiaries in the United States and/or other countries.

Node.js® is an official trademark of Joyent. NVMesh is not formally related to or endorsed by the official Joyent Node.js open source or commercial project

OpenStack® is an official trademark of the OpenStack Foundation.

MONGO and MONGODB are trademarks of MongoDB Inc., registered in the United States and other countries.

Microsoft® and Azure® are trademarks of Microsoft, Inc., registered in the United States and other countries.

## 2. Preface

---

Excelero™ creates innovative, high performance storage solutions that accelerate business applications and deliver outstanding return on investment with the lowest cost of ownership. The NVMesh® software defined block storage product offers Elastic NVMe: the performance of local NVMe server flash with the convenience, efficiency and redundancy of an all-flash-array. For details, go to: [www.excelero.com](http://www.excelero.com).

This document describes how a user can deploy and install the Excelero NVMesh storage solution. For information on product changes and notes refer to [NVMesh Release Notes](#).

### AUDIENCE

The primary audience for this document is intended to be storage administration and application administration personnel responsible for installing and deploying the Excelero NVMesh product.

### NON-DISCLOSURE REQUIREMENTS

© Copyright 2015-2022 Excelero, Inc. All rights reserved. This document contains proprietary information of Excelero, Inc. Do not reproduce or distribute without the prior written consent of Excelero.

### FEEDBACK

We continually try to improve the quality and usefulness of Excelero documentation. If you have any corrections, feedback, or requests for additional documentation, send an e-mail message to [support@excelero.com](mailto:support@excelero.com)

### INFORMATION ABOUT THIS DOCUMENT

All information about this document including typographical conventions, references, and a glossary of terms can be found in the [Document Reference Section](#).

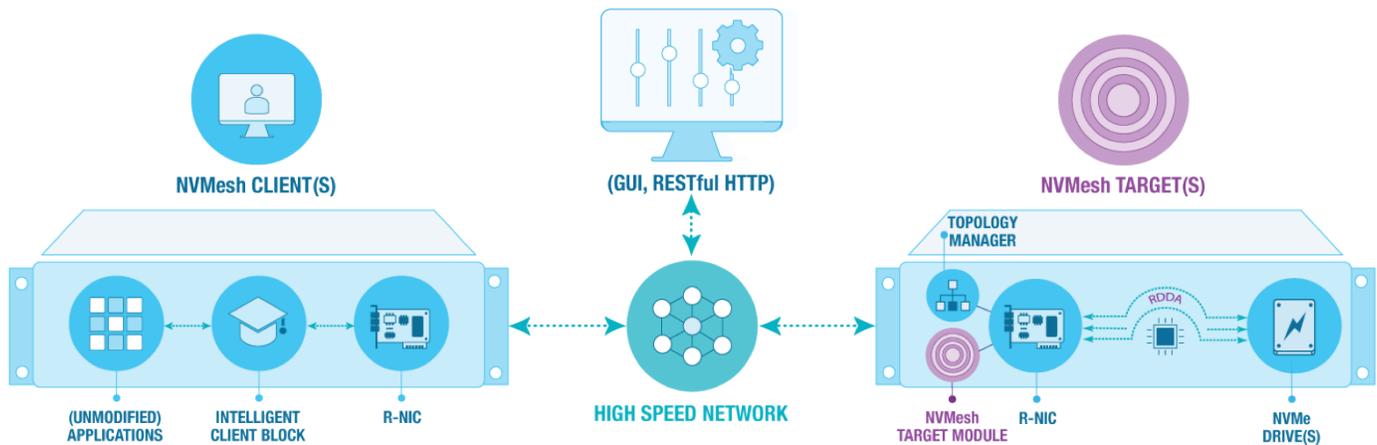
# 3. Introduction To NVMesh Technology

**NVMesh 2.5.2** by Excelero is a high performance, low-latency Software Defined Storage (SDS) product. It provides remote, high speed, low latency storage facilities with NVMe in-server flash performance characteristics utilizing commodity off-the-shelf components. It enables utilization of NVMe drives, potentially spread over many physical systems, as a unified, redundant storage pool. In part, this is accomplished by leveraging Excelero’s patented **Remote Direct Drive Access (RDDA)** functionality. As **NVMesh 2.5.2** is a software only solution, it has the flexibility to provide redundant, centrally managed storage in a hyper-converged architecture or a disaggregated (top-of-rack) storage solution.

## 3.1. Overview

**NVMesh 2.5.2** is comprised of four main software elements:

- Management
- Intelligent Client Block Driver (initiator)
- Target
- **Topology Manager (TOMA)**



*NVMesh Software and Hardware Components*

## Management

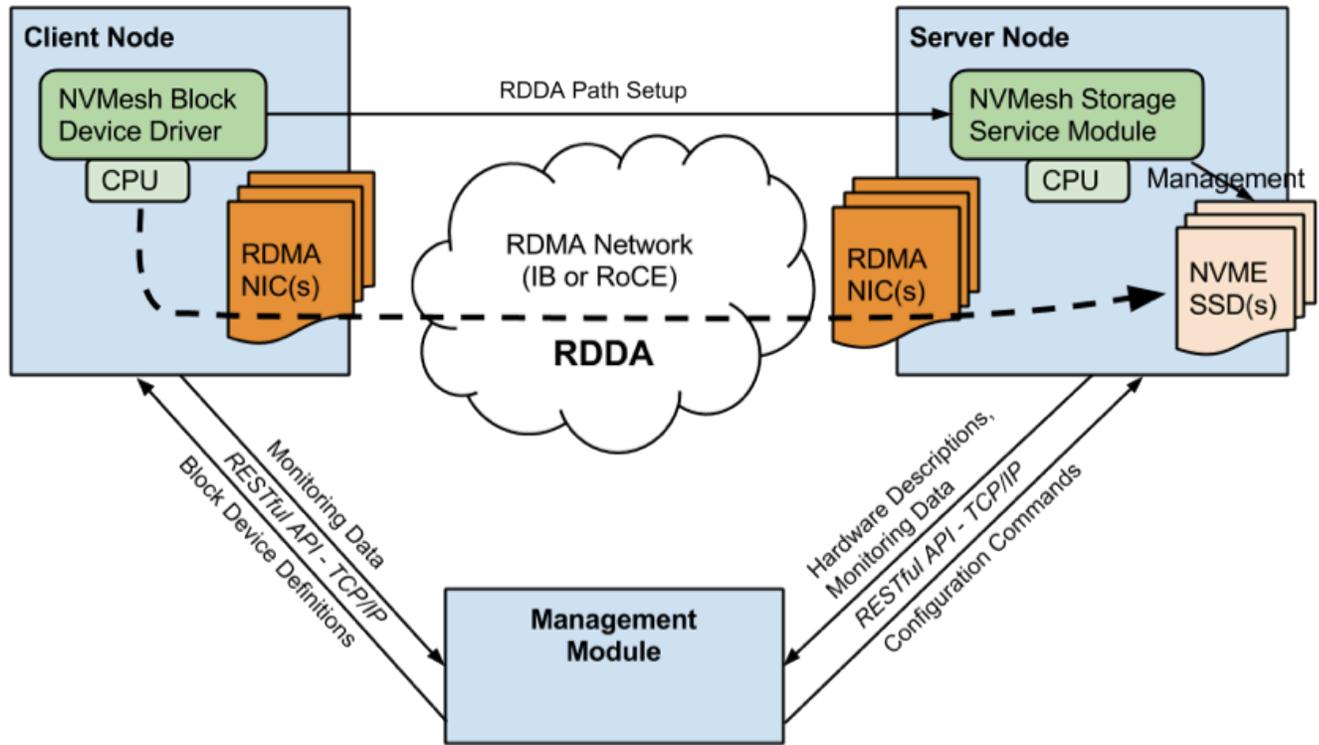
**Management** is the centralized administration element of an **NVMesh 2.5.2** deployment providing storage management, configuration and monitoring functionality. It functions as a web based GUI and as a RESTful API management interface.

## Client Block Driver (Initiator)

The **NVMesh 2.5.2 Client** implements block device functionality for storage clients or consumers, often named initiators. After volumes are attached, they manifest themselves as block devices under the `/dev/n`

vmesh directory (on Linux).

A node that has one or more NVMe devices to share is called a converged node. Converged nodes run both the **Client** and the **Target** software. **Management** can also be run on a **Client** or on a converged node.



NVMesh Component Interactions

## Target

The **Target** identifies storage hardware, NVMe drives and compatible NICs, and sets up **RDDA** pathways into the target drives on behalf of the storage clients. This software runs on a target node, which is any physical node containing solid state drives to be shared. **Targets** also run a user-space software component called **TOMA (Topology Manager)**.

All **Targets** are also **Clients** as they require the client block driver software to be run.

## Topology Managers (TOMA)

The **topology managers** running across the cluster communicate between themselves and form a quorum, with one node elected as the leader.

The leader distributes the topology changes to the other topology managers in the cluster. The managers detect error conditions, such as node reboots or missing drives, and are responsible for error handling activities. It monitors for events like drive or network status changes. It initiates rebuilds and prevents “split brain” in mirrored volume configurations.

## 3.2. Deployment and Nomenclature

It is possible to run one or more **NVMesh 2.5.2** components on a single machine, providing storage system architecture flexibility. A typical setup will comprise several *client nodes* and several *target nodes*. The *management* can run virtually anywhere with TCP/IP connectivity to the *client* and *target nodes*.

For best performance, the *client nodes* and *target nodes* should have one or more RDMA NICs. If there are multiple ports or NICs, **NVMesh 2.5.2** will attempt to use all of them by default. It is possible to limit which NICs will be used. **NVMesh 2.5.2** supports RDMA over Converged Ethernet version 2 (RoCEv2) as well as RDMA over InfiniBand.

A *target node* running the **target module** should have one or more NVMe SSDs. The **target module** supports setting up **RDDA** pathways on RDMA capable NICs. For most kernels, it is required to install the Mellanox OpenFabrics Enterprise Distribution (OFED) RDMA drivers. For more modern kernels, the “inbox” driver is sufficient. Supported NICs and OFED versions can be found in the separate [NVMesh Interoperability Matrix](#). Mellanox OFED drivers are downloadable directly from Mellanox: [Download Mellanox OFED](#)

## 3.3. Target and Client Node Communication

**Client** data path communication travels directly to **Targets**. There is no meta-data manager lookup, redirection or other system bottleneck often associated with legacy SDS solutions. It is important to note that there is no communication between **Clients**. Similarly, for client-initiated IO, **Targets** do not send data path communication to each other.

**Targets** do communicate for rebuilding a failed drive on a peer **Target**. They also perform regular control path communication with each other utilizing a light-weight status or keep-alive protocol.

## 3.4. Software Components

**NVMesh 2.5.2** comprises four software packages:

Package Name	Contents
nvmesh-core	The <b>NVMesh 2.5.2 Clients</b> and <b>Targets</b>
nvmesh-management	The <b>NVMesh 2.5.2 Management Servers</b>
nvmesh-nvmft	A gateway for exposing <b>NVMesh 2.5.2</b> volumes via <b>NVMe-oF</b>

nvmesh-utils	<p>A set of utilities for use with <b>NVMesh 2.5.2</b>:</p> <ul style="list-style-type: none"> <li>• <b>nvmesh</b> – <b>NVMesh</b>'s command-line-interface, see the <a href="#">NVMesh CLI Guide</a></li> <li>• Python SDK libraries, which enable simple automation of Python-based orchestration tools</li> <li>• <b>nvmesh_diag</b> – a tool for collecting server environment information to assist in diagnosing issues and verifying compatibility with <b>NVMesh 2.5.2</b></li> <li>• <b>nvmesh_logs_collector</b> – a utility for collecting system-wide information for <a href="#">Excelero Technical Support</a> to diagnose system issues</li> </ul>
--------------	---

## 3.4.1. Management

**NVMesh 2.5.2 Management** is used to implement management functionality only. Its absence does not affect system block device functionality.

**Management** is a web server based application and can run on virtually any Linux system, including **Targets** or **Clients**. Once installed, the management services can be started and stopped via a system service called `nvmeshmgr`. This software can run on any host with TCP/IP connectivity to **Targets** and **Clients**. Setting up scalable and redundant **Management Servers** is supported.

From a **Client**'s perspective, **Management Servers** are used to provide volume definitions. **Targets** report their inventory of shareable NVMe devices and their performance metrics to the **Management Servers**.

Once installed and configured, **Management** can be started and stopped via a system service called `nvmeshmgr`. This service is dependent upon the `nvmeshstats` service. Both services provide standard service status information. Both services are implemented as node.js applications. The `nvmeshmgr` service is enabled upon installation of the `nvmesh-management` software package, so it will automatically start with the operating system.

 **Management Servers** are only involved in configuration, monitoring and error recovery. They are not involved in the data path, between **Clients** and **Targets**, at any time.

It is possible to run **Management Servers** in a container. Consult with [Excelero Technical Support](#) for details.

## 3.4.2. Target

The main **Target** components are a kernel module and **TOMA** (Topology Manager). The **Target** software should be run on any node with one or more drives to be pooled. Very small **NVMesh 2.5.2** deployments may utilize nodes without NVMe devices for purposes of establishing **TOMA** quorum.

**Targets** report the hardware, i.e. NVMe devices and NICs, status and performance statistics to

**Management Servers.** They receive configuration information from **Management Servers**, including volume definitions and the coordinates of the other **Targets**. They receive requests from **Clients** for attaching volumes and if deemed valid, set up direct **RDDA** connectivity between the **Clients** and the drives. **Targets** also set up non-RDDA client-to-target communication paths to implement additional data path functionalities.

The target software also includes a user-mode component named **TOMA**. The **TOMA** process receives up-to-date configuration information and is responsible for monitoring volumes for error conditions, ensuring data integrity, avoiding “split brain” conditions and coordinating volume rebuild activities. **TOMA** also communicates light-weight keep-alive information to other TOMAs. In the case of a volume recovery or rebuild, **TOMA** orchestrates the rebuild operation. Only during this failure recovery operation may **Target** to **Target** communication be intensive.

**NVMesh 2.5.2 Targets** comprise a Linux kernel loadable module named `nvmeibs`. **TOMA** is implemented by a user-mode process named `nvmeibt_toma`.

**!** Note: **NVMesh 2.5.2** kernel modules are compiled for specific combinations of operating system, kernel and OFED version. Care must be taken when upgrading the kernel or the version of OFED. Changing the kernel or OFED drivers to unsupported versions may result in **NVMesh 2.5.2** becoming nonfunctional. Supported operating system, kernel and OFED versions can be found in the [NVMesh Interoperability Matrix](#).

Once installed and configured, the storage services can be started and stopped via a system service called `nvmeshtarget`. This service is dependent on the `nvmeshclient` service. It also requires the `nvmeshtoma` a service that manages the **TOMA** process. Both services provide standard service status information. The `nvmeshtarget` service is disabled by default upon installation of the `nvmesh-core` software package so it will not start with the operating system. Run `systemctl enable nvmeshtarget` to alter this.

## 3.4.3. Client

---

**NVMesh 2.5.2** provides a block device interface to storage clients. The block device interface is implemented by the **Client** software.

Upon receiving an `nvmesh_attach_volumes` command, the **Client** communicates with **Management** to obtain the requested volume’s configuration and topology. It then communicates directly with the appropriate **Targets**, forming connections to the **Target** NVMe devices that make up the volume. Finally, it creates a generic block device under `/dev/nvmesh/<volume-name>`. From that point on, any IO utilizing the block device travels directly to and from the NVMe devices that make up the volume on **Targets**.

The client software comprises multiple Linux kernel loadable modules as follows. All modules except `nvmeibc` and `nvmeiba` are also utilized by the **Target**’s kernel module `nvmeibs` and are considered “common” modules.

Kernel Module	Function
<code>nvmeibc</code>	<b>NVMesh 2.5.2</b> block device implementation
<code>nvmeiba</code>	<b>NVMesh 2.5.2</b> block device shim enabling live software upgrade
<code>nvmeib_common</code>	<b>NVMesh 2.5.2</b> private kernel calls
<code>nvmeib_common_public</code>	<b>NVMesh 2.5.2</b> public calls
<code>nvmeib_common_siw_public</code>	Soft iWarp public calls
<code>nvmeib_public_bnxt_re</code>	Broadcom NIC specific calls
<code>nvmeib_common_mlx5_public</code>	Mellanox NIC specific calls

**!** Note: **NVMesh 2.5.2** kernel modules are compiled for specific combinations of operating system, kernel and OFED version. Care must be taken when upgrading the kernel or the version of OFED. Changing the kernel or OFED drivers to unsupported versions may result in **NVMesh 2.5.2** becoming nonfunctional. Supported operating system, kernel and OFED versions can be found in the [NVMesh Interoperability Matrix](#).

Once installed, the **Client** services can be started and stopped via a system service called `nvmeshclient`. The `nvmeshclient` service depends on two services for communicating with **Management** for reporting its status and performance statistics, `nvmeshcm` and `nvmeshagent` that run a python process each named `managementCM.py` and `managementAgent.py` respectively. All services provide standard service status information. The `nvmeshclient` service is enabled upon installation of the `nvmesh-core` software package, so it will automatically start with the operating system.

### 3.4.3.1. Multiple Client Instances

It is possible to run multiple **Client** instances on a single node. Each instance can have a different configuration enabling functionality not available from a single instance. Each instance can have a logical name associated with it for ease of administration. Use the `nvmesh_client_instance_do` tool to manage the instances.

The default prefix for **NVMesh 2.5.2** block devices is `/dev/nvmesh`. Alternative prefixes, including the option to use the default `/dev` directory, are available per **Client** instance. When invoking the `nvmesh_client_instance_do` tool to generate a new instance, the optional `--blkdir` parameter can be used to define an alternative location for block devices.

In the future, having per-instance configurations will enable associating different **Client** instances to different **NVMesh** clusters. Per-instance configurations will also enable defining networking resources per instance.

**Note:** With *NVMesh 2.5.2*, this functionality is still in implementation and is not made generally available.

## 3.4.3.2. NVMe-over-Fabrics Gateway

---

*NVMesh 2.5.2* can be used to expose volumes via an NVMe-over-Fabrics (*NVMe-oF*) gateway. This option enables accessing *NVMesh 2.5.2* storage without installing the *Client*. This can be employed when using *NVMesh 2.5.2* as a storage option for multiple operating systems and virtualization frameworks not supported currently by *NVMesh 2.5.2*.

The gateway is packaged in the `nvmesh-nvmft` software package. The *NVMe-oF* gateway functionality is globally enabled via the advanced general settings in the *Management* GUI. A plug icon adjacent to the volume name in the *Clients* section in the GUI indicates that the gateway exposes the volume.

For each volume, there is an option to enable *NVMe-oF* access to it and to define the set of *Clients* that can be used as gateways. To make a *Client* defined in the volume's set act as a gateway ensure that the `nvmesh-nvmft` package is installed on the client node. Next, ensure that the `nvmeshnvmft` service is enabled and running. By default, the service will be enabled, i.e. set to run upon operating system startup, but not running yet upon the initial installation. Thereafter, once the volume is attached to this *Client*, it will be accessible from *NVMe-oF* initiators. Each volume will be exposed under a distinct *NVMe-oF* controller with a single namespace. The namespace's global id will be that of the volume. Therefore, it should be distinguishable regardless of the *Client* exposing it.

The gateway is implemented by a process named `nvmf_tgt`. It is set to run on the first 12 cores of the *Client*. These cores will not be available for other workloads in practice. Contact [Excelero Technical Support](#) for guidance on changing this pending more elaborate configuration via *NVMesh*.

The *NVMe-oF* implementation supports both the RDMA and TCP networking options. It will use all networking interfaces by default. Contact [Excelero Technical Support](#) for guidance on altering the network configuration.

The NQN for a volume via NVMe-oF will be `nqn.2016-016.io.spdk:<VOLNAME>_subsys_<CLIENT_HOSTNAME>`. For instance, a volume named `vol1` exposed via *NVMe-oF* from a client named `client1.acme.com` will have an NQN of `nqn.2016-06.io.spdk:vol1_subsys_client1.acme.com`.

**Note:** The term *nvmf* is used as a synonym for *NVMe-oF*.

**Installation Note:** On Ubuntu systems, it may be required to install the `linux-modules-extra` package for the current kernel for the the gateway to function properly. Adding this as a dependency is under consideration.

## 3.5. High Availability

---

*NVMesh 2.5.2* provides out of the box high availability for both software and hardware components.

## Server Availability

**NVMesh 2.5.2** constantly monitors the availability of the cluster nodes. When a target server fails due to a hardware or software problem, within a few seconds the other target servers will identify the failure and switch the relevant volumes to degraded as necessary.

## Network Availability

**NVMesh 2.5.2** constantly monitors the availability of network links, both in the clients and targets. Links are failed over and back automatically.

## Drives

In a protected volume, drives may fail and the missing contents rebuilt on alternate capacity via mirrored copies or parity calculation, depending on protection method.

## 3.6. Load Balancing

---

By default, **NVMesh 2.5.2 Clients** leverage all available network links or paths between themselves and the **Targets**. Traffic is aggregated over multiple links. If a link fails, the **Client** will automatically divert traffic to another available link.

 When a link fails and becomes active again, it may take up to 30 seconds before the client will start using this link.

## 3.7. Drive Format Types

---

### Overview

Drives must be formatted before use. **NVMesh 2.5.2** supports two format types:

- 4k+8 formatting. 4k is the size of data in each drive block. 8 is the number of metadata bytes per drive block.
- 4k+0 formatting

### 4k+8 Formatting

This formatting supports all **NVMesh 2.5.2** volume types, including erasure-coded or parity-based volumes. This format type is only available on drives supporting 4K+8 bytes sectors with *Metadata Pointer Support*.

See the [NVMesh Interoperability Matrix](#) for more details. During the formatting of a drive to this format, all blocks are zeroed.

The following describes the on-disk structure of such formatting:

MBR	4K
GPT Header	4K
GPT Table	20K
Excelero Private Area	Up to ~0.5% of the drive's capacity
Journal	2GB
Journal Metadata	32MB
...	
Data	
...	
GPT Table – second copy	20K
GPT Trailer	4K

## Checking for 4k+8 Format Compatibility

To check whether a drive can be used for 4k+8 formatting, use the `nvme` command-line tool supplied with the `nvme-cli` package in the supported operating system distributions. Perform the following steps as root.

Run `nvme list` to get a list of all devices on the node. Identify the device name from the Node column for the device to be checked, usually this can be done using the SN (serial number) or its Model.

For a device with a block device name of `/dev/nvme1001`, run `nvme id-ns -H /dev/nvme1001`.

The last lines of output contain the supported block formats. If there is a line similar to the following with a Metadata size of 8 bytes and a Data size of 4096 bytes exists, this drive supports 4k+8 bytes sectors.

```
LBA Format 3 : Metadata Size: 8 bytes - Data Size: 4096 bytes - Relative Performance: 0x2 Good (in use)
```

To verify support for *Metadata Pointer Support*, locate the `mc` section of the output from that same command. If it is supported, it should appear as follows with the wording “Metadata Pointer Supported”.

```
mc          : 0x3
  [1:1]    : 0x1      Metadata Pointer Supported
  [0:0]    : 0x1      Metadata as Part of Extended Data LBA Supported
```

## 4k+0 Formatting

This formatting supports all volume types except erasure-coded volumes. This format type is used only for drives that do not support 4K+8 bytes sectors. Drives with 4k+8 bytes sectors, but without *Metadata Pointer Support* are not supported in NVMesh in general.

## Formatting Timeouts

Formatting operations take a few seconds to a few minutes, depending on the size of the drive and the performance of TRIM operations on it, sometimes referred to as erase operations or deallocations. There is a default format timeout determined by the `nvmeibs` module parameter, `submit_wait_timeout`, set to 15 seconds. For each drive individually, the format timeout will be increased by this value upon a failed format. See the [Configuration Options](#) section for more details on managing parameters.

Most drives format within a few seconds, but some drives, especially with larger capacity, may require more time.

## 3.8. Logical Volume Types

---

**NVMesh 2.5.2** supports multiple volume types that vary in their data layout and their level of data protection. Currently supported logical volume types include:

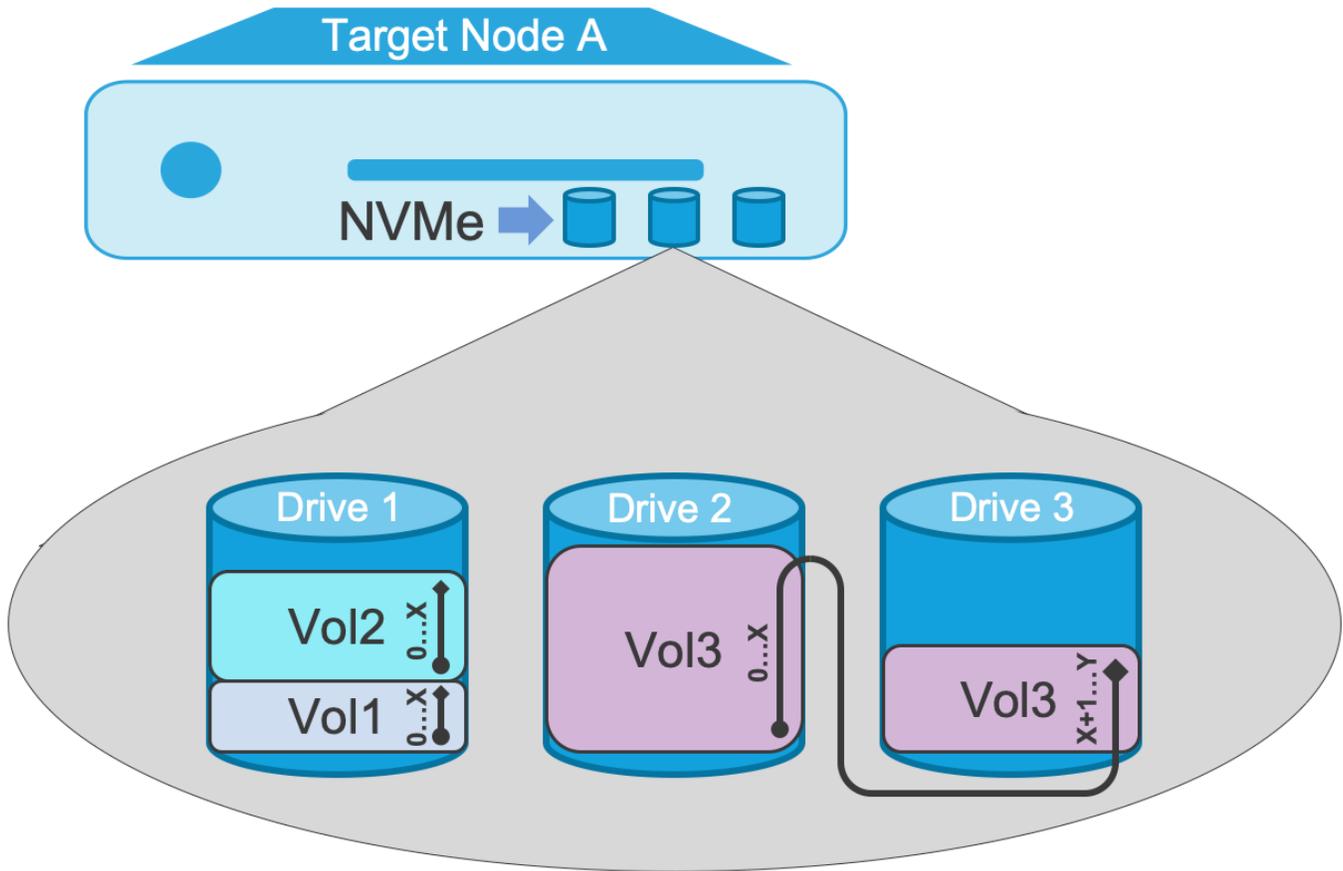
- Concatenated
- Striped RAID-0
- Mirrored RAID-1
- Striped & Mirrored RAID-10
- Erasure-coded, often also called parity-based

Volumes are allocated from the pool of storage devices managed by **NVMesh 2.5.2**. Volumes may utilize a portion of one device or portions of multiple devices. A portion of a device may also be an entire device. A single device may host a portion of a single volume or may host portions of multiple volumes. A portion of a volume may also be the entire volume.

### 3.8.1. MeshProtect Concatenated

---

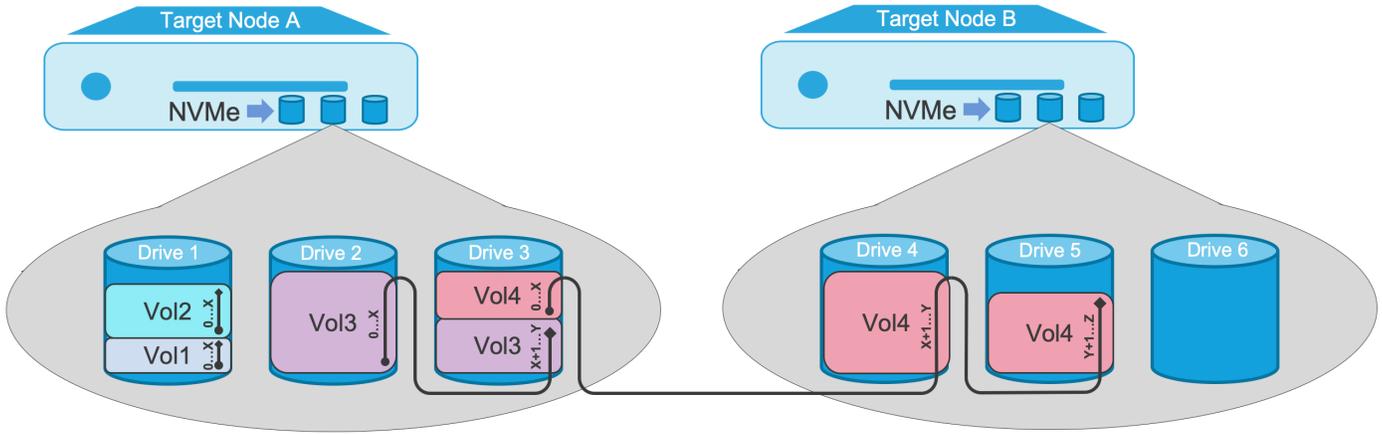
## MeshProtect Concatenated Logical Volumes, Single Target Node



*Concatenated volumes on a single target node*

- Vol1 = Concatenated volume contained within a single drive. LBAs are mapped to a contiguous range on the single drive. Performance of the volume is limited to the performance of the 1 drive.
- Vol2 = Concatenated volume similar to Vol1, but a larger size, about 500GB.
- Vol3 = Concatenated volume whose size specification was larger than a single drive. LBAs are contiguous across Drive 2 and then continue on Drive 3. Sequential IO is limited to the performance of 1 drive, regardless of how many drives make up the volume. Random (4K) read/write performance across the entire volume LBA range is equivalent to about 2 drives.
- Volume capacity overhead is 0% – volume size is equivalent to aggregate segment allocation.
- Concatenated volumes offer no protection against failures. If any drive (or host) that is part of a concatenated volume is unavailable, the volume will go offline. A failed drive results in permanent data loss.

## MeshProtect Concatenated Logical Volumes, Multiple Target Nodes

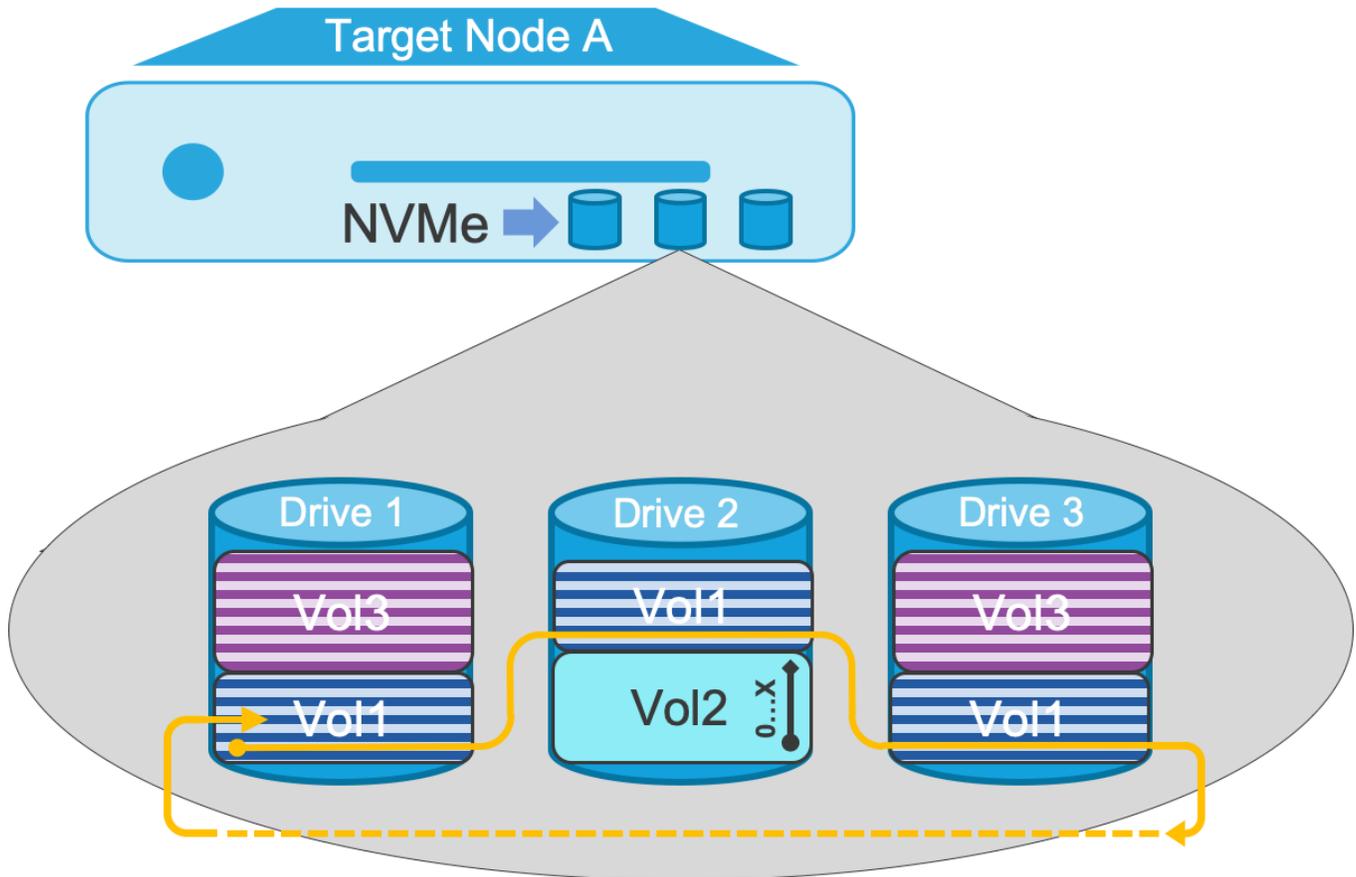


*Concatenated volumes across multiple target nodes*

- Vol1, Vol2 and Vol3 are the same as in the previous example.
- Vol4 = Concatenated volume whose size specification was larger than 2 drives. LBAs are contiguous across a portion of Drive 3 and then continue across Drive 4 and on Drive 5. Sequential IO is limited to the performance of 1 drive, regardless of how many drives make up the volume. Random (4K) read/write performance across the entire volume LBA range is equivalent to about 3 drives.
- Volume capacity overhead is 0% – volume size is equivalent to aggregate segment allocation.
- Concatenated volumes offer no protection against failures. If any drive (or host) that is part of a concatenated volume is unavailable, the volume will go offline. A failed drive results in permanent data loss.

## 3.8.2. MeshProtect 0 (Striped RAID-0)

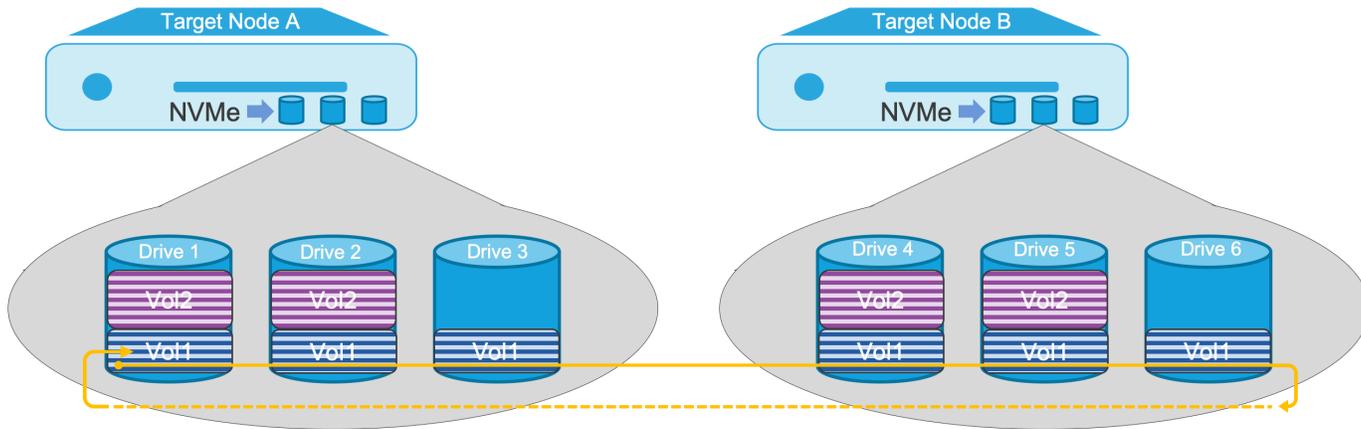
### MeshProtect 0 Logical Volumes, Single Node



*MeshProtect 0 Volumes on a single node*

- Vol1 = MeshProtect 0, 3 wide. LBAs are spread across repeating stripes on Drives 1, 2 and 3, writing  $32 \times 4K$  blocks (128K) before going to the next drive. While the segment size on each drive is the same, the segments do not need to be in the same range on each drive. Sequential performance of the single volume is the aggregate performance of all 3 drives. Random (4K) read/write performance across the entire volume LBA range is equivalent to about 3 drives.
- Vol2 = Concatenated, capacity smaller than Drive 2. This simply demonstrates that segments from different types of Volumes can be on the same drive.
- Vol3 = MeshProtect 0, 2 wide. Sequential performance of the single volume is the aggregate performance of 2 drives. Random (4K) read/write performance across the entire volume LBA range is equivalent to about 2 drives.
- Volume capacity overhead is 0% – volume size is equivalent to aggregate segment allocation.
- MeshProtect 0 (Striped) volumes offer no protection against failures. If any drive (or host) that is part of a concatenated volume is unavailable, the volume will go offline. A failed drive results in permanent data loss.

## MeshProtect 0 Logical Volumes Across Multiple Nodes



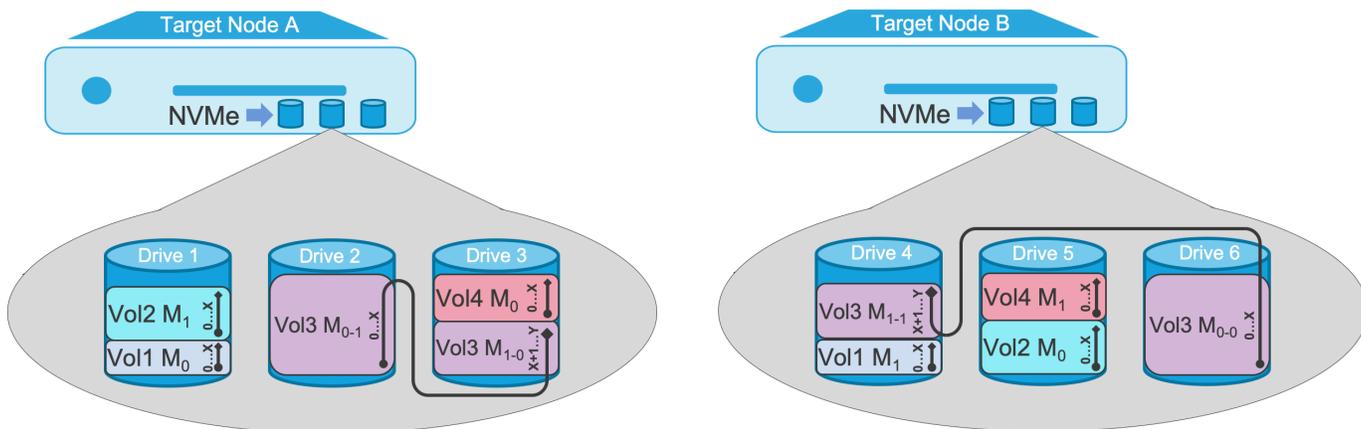
*MeshProtect 0 volumes across multiple nodes*

- Vol1 = MeshProtect 0, 6 wide, over 2 target nodes. LBAs are spread across repeating stripes on 6 drives, writing  $32 \times 4K$  blocks (128K) before going to the next drive. If target nodes had only a single 100GbE interface, 4 drives would max out the interface. By spreading over 2 target nodes, sequential performance of the single volume is the aggregate performance of all 6 drives. Random (4K) read/write performance across the entire volume LBA range is equivalent to about 6 drives.
- Vol2 = MeshProtect 0, 4 wide. Sequential performance of the single volume is the aggregate performance of 4 drives. Random (4K) read/write performance across the entire volume LBA range is equivalent to about 4 drives.
- Volume capacity overhead is 0% – volume size is equivalent to aggregate segment allocation.

### 3.8.3. MeshProtect 1 (Mirrored)

## MeshProtect 1 Logical Volumes

A MeshProtect 1 (Mirrored) volume consists of an exact copy or mirror of a set of data on two or more devices. If a capacity amount is specified that is larger than the devices, more devices will be concatenated to the volume in mirrored pairs serially, as in the description of concatenated volumes.

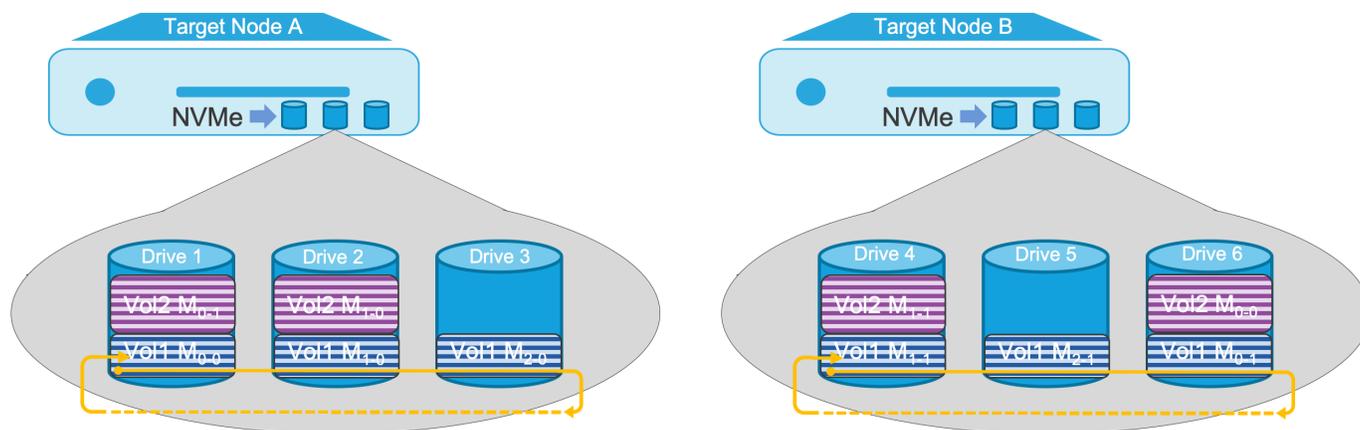


*MeshProtect 1 volumes across multiple nodes*

- Vol1 = MeshProtect 1 over 2 target nodes and only 2 drives. Mirror segment pairs are allocated on different target nodes. Volume size fits within 1 drive and segment allocations are mirrored between Drive 1 and 4. Sequential read performance is equivalent to 2 drives as reads alternate between mirrors. Sequential write performance is limited to a single drive as both mirrors are written synchronously. Random (4K) read performance is equivalent to 2 drives. Random (4K) write performance is equivalent to 1 drive.
- Vol2 = MeshProtect 1 similar to Vol1. Defined volume size is smaller than a single drive. Mirrored segments are the same size but at different ranges on Drive 1 and 5.
- Vol3 = MeshProtect 1, over 2 target nodes, 4 drives. Defined volume size exceeds a single drive. LBAs are contiguous and mirrored across Drive 2 and Drive 6, then continue mirrored between Drive 3 and 4. Mirrored segments are the same size but can be on different ranges on different drives. Sequential read IO performance is the aggregate of 2 drives regardless of how many drives make up the volume. Sequential write performance is limited to a single drive. Random (4K) read performance across the entire volume LBA range is equivalent to about 3 drives. Random (4K) read performance on limited LBA ranges is the equivalent of 2 drives. Random (4K) write performance across the entire volume LBA range is equivalent to about 1.5 drives. Random (4K) write performance on limited LBA ranges is the equivalent of 1 drive.
- Volume capacity overhead is 100% – volume size is equivalent to 50% of the allocated segments.
- These volumes can remain online if any one drive fails, or either target host fails. In this degraded mode, write performance is the same while read performance is cut in half.

### 3.8.4. MeshProtect 10 (Striped & Mirrored)

#### MeshProtect 10 Logical Volumes



*MeshProtect 10 volumes across multiple nodes*

- Vol1 = MeshProtect 10, 3 wide, over 2 target nodes. LBAs are spread across repeating stripes on mirrored pairs of segments across sets of 3 drives (6 drives total), writing 32K before going to the next drive. Segments on Drive 1, 2 and 3 are mirrored to Drive 6, 4 and 5. Sequential and Random (4K) write performance is equivalent to 3 drives as writes are synchronously written to mirrored segment pairs. Sequential and Random (4K) read performance is equivalent to 6 drives as reads are alternated

between mirrors.

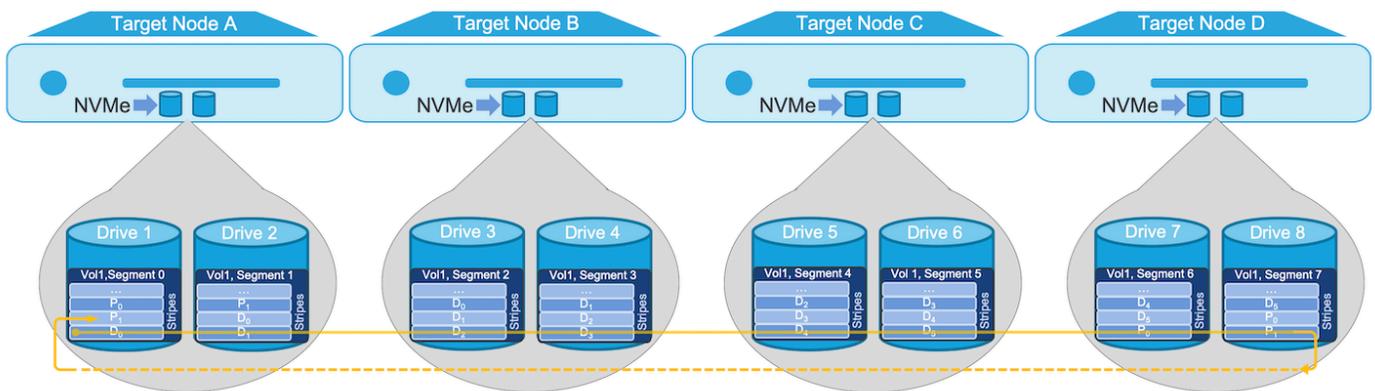
- Vol2 = MeshProtect 10, 2 wide. Sequential and Random (4K) write performance of the single volume is the aggregate performance of 2 drives. Sequential and Random (4K) read performance of the single volume is the aggregate performance of 4 drives.
- Volume capacity overhead is 100% – volume size is equivalent to 50% of the allocated segments.
- Stripes utilize a 128K chunk; writing 32 × 4K blocks (128K) to both mirrored drive segments before going to the next drive.
- These volumes can remain online if any one drive fails, or either target host fails. In this degraded mode, write performance is the same while read performance is reduced by the loss of the failed drive.

### 3.8.5. MeshProtect 60 (Dual Parity)

A MeshProtect 60 volume is a volume that stores data and parity across multiple devices. It uses **distributed erasure coding**, a method of data protection in which data blocks and redundant parity blocks are distributed across a set of devices, forming one protected logical storage unit. A MeshProtect 60 volume provides N+2 data protection, meaning that one or two devices may fail but the volume will still be intact.

The examples below are for MeshProtect 60, but MeshProtect 50 (Single Parity) is nearly identical with the main difference being only 1 parity calculation per stripe, allowing for only a single drive or host failure.

## MeshProtect 60 (6+2) Logical Volumes Across Minimum Number of Target Nodes

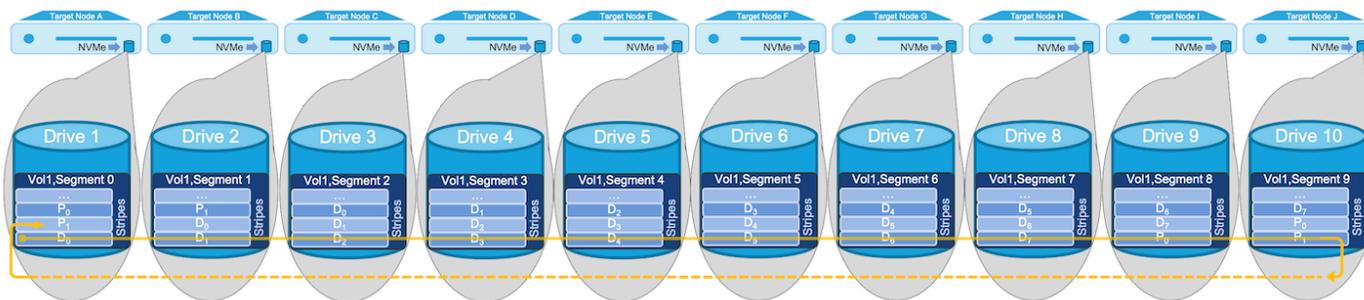


*MeshProtect 60 (6+2) volumes across multiple nodes*

- Vol1 = MeshProtect 60, 6+2, Dual-Parity over 4 target nodes. LBAs are spread across repeating stripes of segments across 8 drives total. Each stripe consists of 6 data chunks and 2 parity chunks. The parity chunks are rotated over the segments to avoid drive wear due to read-modify-write operations. Sequential and Random (4K) read performance is equivalent to the aggregate of all drives. Sequential and Random (4K) write performance is heavily dependent on the drives, the initiator host and the network, but is roughly equivalent to 3-4 drives.

- Volume capacity overhead for this example (6+2) 33% – volume size is equivalent to 75% of the allocated segments. MeshProtect 60 is variable with 8+2 being common at 25% overhead, or 80% useable drive capacity.
- MeshProtect volumes guard against both drive and host failures. The minimum number of hosts needed to create a volume and survive a host failure is calculated by taking the number of data segments (6), plus parity segments (2) and divide that result by the number of parity segments. Example:  $(6+2)/2 = 4$  target nodes minimum. You would need 8 target nodes for 6+2 to have dual node redundancy.
- In this 6+2 example, any 1 host can fail, or any two drives can fail and the volume will remain online.

## MeshProtect 60 (8+2) Logical Volumes Across Target Nodes with Dual Node Failure Redundancy



MeshProtect 60 (8+2) volumes across 10 nodes

- Vol1 = MeshProtect 60, 8+2, Dual-Parity over 10 target nodes. LBAs are spread across repeating stripes of segments across 10 drives total. Each stripe consists of 8 data chunks and 2 parity chunks. The parity chunks are rotated over the segments to avoid drive wear due to read-modify-write operations. Sequential and Random (4K) read performance is equivalent to the aggregate of all drives. Sequential and Random (4K) write performance is heavily dependent on the drives, the initiator host and the network, but is roughly equivalent to 4-5 drives.
- Volume capacity overhead for this example (8+2) 25% – volume size is equivalent to 80% of the allocated segments.
- In this 8+2 example, any 2 hosts can fail, or any two drives can fail and the volume will remain online.
- Of course, you can have many more than a single drive per target node.

## Supported Data plus Parity Combinations

The data and parity configuration options are listed in the following table.

Data Drives	Parity Drives
2, 3, 4, 5, 6, 7, 8, 9, 10 or 11	1

2, 3, 4, 5, 6, 7, 8, 9, 10	2
----------------------------	---

The amount of data written per device before moving to the next device in the stripe, is currently fixed at 4K. For an 8+2 configuration, the total amount of data in a stripe is then 32K with an additional 8K used for parity.

**Note:** Trim operations are ignored for erasure coded volumes.

## 3.8.5.1. Target Node Redundancy

MeshProtect 60 (Dual Parity) *volumes* can be configured in various *target* redundancy levels:

- N+2 Target Redundancy – Only one *volume* segment per *target node*. In this mode, the *volume* can survive up to two *target* failures (two drive segments can fail concurrently).
- N+1 Target Redundancy – Up to two *volume* segments per *target node*. In this mode, the *volume* can survive a single *target* failure (one drive segment can fail).
- No Target Redundancy – No restriction on *volume* segments per *target node*. In this mode, the *volume* will not survive even one *target* failure.

MeshProtect 1 and 10 guard against a single drive failure, or a single *target node* failure.

MeshProtect 60 *volumes* can guard against both drive and host failures. The minimum number of *target* hosts needed to create a *volume* and survive a host failure is calculated by taking the number of data segments (D), plus parity segments (P) and divide that result by the number of parity segments (D+P)/P. Example: (6+2)/2 = 4 target nodes minimum. You would need 8 *target nodes* for 6+2 to have dual node redundancy.

### Minimum Number of Target Nodes

Data + Parity Drives	Dual Target Node Redundancy	Single Target Node Redundancy	No Target Redundancy
6+2	8 Target Nodes	4 Target Nodes	1 Target Node
8+2	10 Target Nodes	5 Target Nodes	1 Target Node

## 3.9. Access Modes

**NVMesh 2.5.2** volumes are inherently shared read-write volumes enabling multiple **Clients** to attach to the same volume. However, for some use cases, it is useful to ensure the volume is being accessed in the right manner across all clients. A volume can only be accessed with a single access mode across the entire cluster and this is considered the volume’s current mode. If a volume is not attached to any **Client**, it may be in no mode currently. This also applies to multiple **Client** instances running on the same physical node.

Trim operations are considered write operations from the access mode perspective.

**NVMesh 2.5.2** supports the following access modes:

- **Shared Read Write** – This is the default access mode. With this mode employed, all **Clients** that have the volume attached can simultaneously read and write into it.
- **Exclusive Read Write** – A **Client** that has the volume attached can read and write into it with the assurance that no other **Client** will access it as well. This is useful for using a local file system in a dynamic environment, such as a container oriented one.
- **Shared Read Only** – With this mode employed, all **Clients** that have the volume attached can only read from it. This is useful for accessing a local file system from multiple clients in read-only mode, which provides high performance, with the assurance that no **Client** will write into it.

Volumes are created without an access mode. The access mode is defined once the first **Client** has attached the volume or whenever a client attaches and the volume is without an access mode. When a **Client** detaches when the volume is in the **Exclusive Read Write** mode, it reverts to being without an access mode.

When a **Client** attempts to attach to a volume in a mode different than its current one, the attachment will fail. However, attempting to attach in **Exclusive Read Write** mode will implicitly preempt one of the shared modes. Preemption is needed to move out of a shared mode and in cases where the holder of the exclusive mode did not detach but mode change is required. A preemption may also be needed for one **Client** to forcefully gain access in **Exclusive Read Write** mode whenever another **Client** is attached in this mode.

If a **Client** is attached to a volume and another one preempts it to alter the access mode, it will remain attached, but all outstanding and any additional I/O will fail upon the preemption. To restore IO access, it will be necessary to detach and reattach the volume.

## 3.10. Zones

---

With a standard **NVMesh 2.5.2** installation, all **Drives** are pooled in a single pool. This limits system scalability to some degree, mainly in terms of the total number of **Targets** and the total number of volumes supported. To relieve these limitations, a concept of **Zones** has been introduced from **NVMesh 2.0**. **Zones** are optional. Once this optional functionality has been turned on, a **Target** must be assigned to a **Zone** after it is initially turned on. A zone is identified by a user chosen positive integer.

Zoning functionality can be toggled on/off via *General Settings*.

With zoning on, a **Zone** column is added in the **Targets** table in the **Targets** section of the GUI. If zoning is turned on after volumes have been assigned to a **Target**, but prior to turning on zoning, the **Target** will automatically be assigned to zone 1. It is not permitted to re-assign a **Target** to a different **Zone** after it has been assigned. The only way to do this in practice is to remove the **Target** by evicting or relocating all its **Drives**, deleting it from a **Management** and then restarting the **Target**. This will make it appear as a new zone-assignable **Target**. The **Target** will initially appear to not be in any zone. Choose one or more such

**Targets** and click *Approve* to assign them a zone. A zone is characterized by a zone number, which is chosen in the *Approve* dialog.

Volumes comprise one or more chunks of logical block addresses. Each chunk implements the volume type by spreading the stored data onto *Drives* from one or more **Targets**. Chunks are limited to a single *Zone*. A large volume could be comprised of several chunks, each in a different zone. However, an individual chunk will not span *Zones*. This limitation also applies when the system needs spare space to rebuild a volume.

! When working with *Zones* and relocating a *Drive*, it is imperative to move to a different **Target** in the same *Zone*.

Zones are enumerated from 0 and up. Currently, there is an underlying assumption that all zones up to the maximum exist and volume chunks are allocated randomly from a zone with preference to lesser used zones. Therefore, use zones enumerating them from 0 one at a time. If a single zone is full, allocations may begin to fail for lack of space.

## 3.11. Minimum Configurations

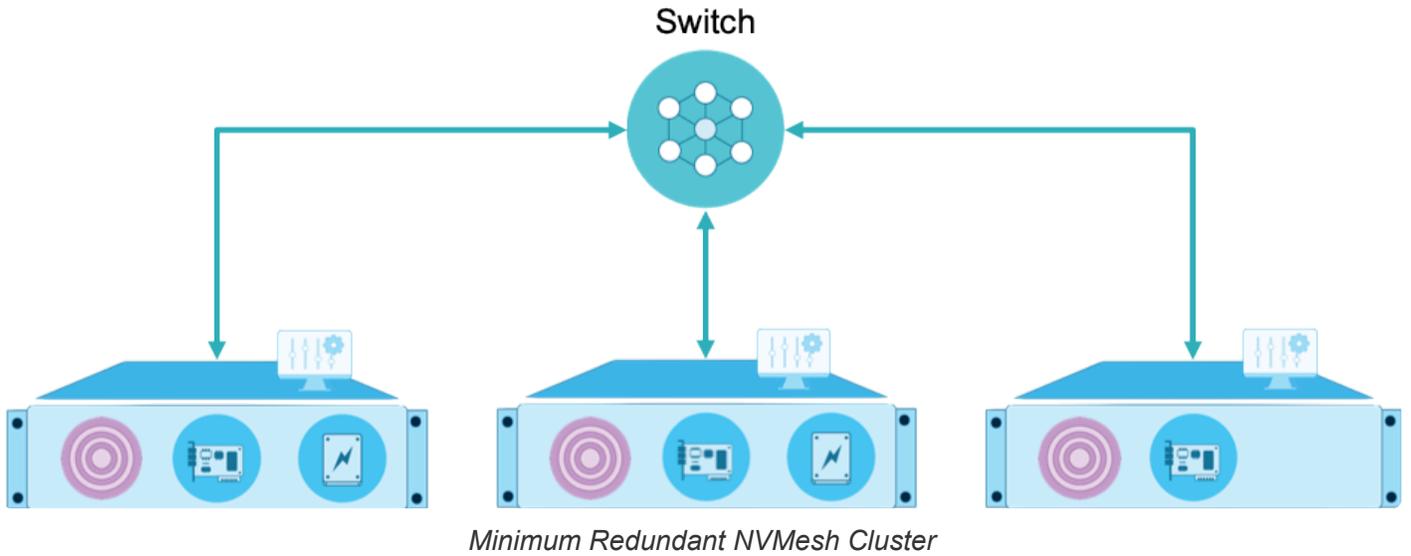
---

### Minimum Cluster for MeshProtect 1/10

The **NVMesh 2.5.2** cluster uses segments spread over drives and hosts to reach a majority consensus on volume statuses. For a MeshProtect 1/10 volume, the minimal number of data drives is two, but a third drive must participate as an arbiter, with a small metadata segment. Because these segments also need to be on separate hosts, the strict minimum redundant **NVMesh 2.5.2** cluster configuration is three hosts. In cases where there are not enough hosts or drives in the cluster to reach majority, a virtual or “dummy” device can be created on a server without any NVMe drives. This device is called an *arbiter drive*. This *arbiter drive* will be created as a file in `/var` on a **Target** but will only contain metadata used for volume consensus.

**Management** can be deployed on a single host. If redundant **Management Servers** are desired, they should be deployed on more than one node. **Management** stores its data in a MongoDB database. To provide redundancy for the database, typically 3 nodes are used.

This results in the minimum redundant **NVMesh 2.5.2** cluster being at least 3 hosts, with at least 2 hosts having physical NVMe drives:



Symbol	Meaning
	NVMesh Management Software
	NVMesh Target Software
	High-speed NIC(s)
	NVMe Drive(s)

*NVMesh Diagram Legend*

To create an arbiter drive, which is needed only if just 2 hosts have NVMe drives, issue the following command on the node without NVMe drives:

```
sudo nvmesh_target add arbiter
```

then restart the **NVMesh 2.5.2 target** service, issue:

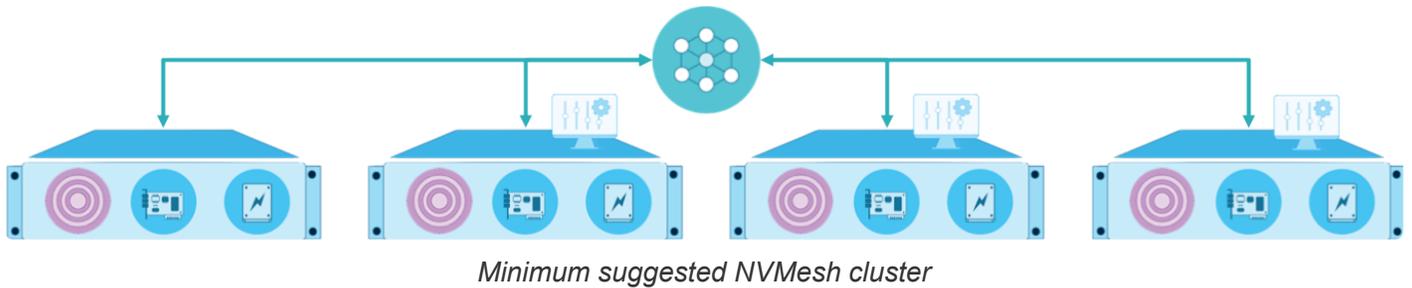
```
sudo systemctl restart nvmeshtarget
```

For example, when running as root:

```
[root@node1 11:19:21 lib]# nvmesh_target add arbiter
Added arbiter /var/opt/NVMesh/metadata_disk_image/241022cd-105a-46fa-aadb-2648c626f099
[root@node1 11:19:21 lib]# systemctl restart nvmeshtarget
```

## Minimum Suggested Cluster for MeshProtect 1/10

The preferred and **suggested** minimum **NVMesh 2.5.2** cluster is 4 hosts, all with NVMe drives, as this allows the creation of MeshProtect 1/10 volumes in balanced host pairs.



In this configuration, there is no need to create an *arbiter drive*.

## Single Node System

In situations where there is no need for high availability, it is possible to install the **NVMesh 2.5.2** client, target and management all on the same server. In this configuration, the administrator can create concatenated or MeshProtect 0 striped volumes. The administrator can also create MeshProtect 60 volumes with drive redundancy, but no node redundancy.

## Minimum Suggested Cluster for Node Redundant MeshProtect 60

For a redundant **NVMesh 2.5.2** system with MeshProtect 60 volumes, at least 3 nodes are needed for the **Management's** MongoDB database, which can also run **Management** and may be converged with **Targets**. The actual number of required **Targets** is dependent upon the MeshProtect 60 parameters chosen and the level of redundancy required. For a detailed expose of this, see the [Target Node Redundancy](#) section.

## 3.12. Security

---

### 3.12.1. Securing Volume Attachments

---

By default, securing volume attachments is disabled in **NVMesh 2.5.2**. To enable this functionality, in the **General** sub-section of the **Settings** section of the GUI, there is a toggle option for **Multi-tenancy** in the **Security** domain.

Once this is enabled, all subsequent attachments are verified. For the attachment to succeed on a **Client**, it must have one of the keys associated with the volume being attached.

*Key Pairs* are managed from the *Key Pairs* sub-section of the *Settings* section. To transfer a *Key Pair* to a **Client**, download it from the management and copy the file generated to `/etc/nvmesh/keys` on the clients, which should be permitted to attach volumes associated with the *Key Pair*.

To associate volumes with a *Key Pair*, generate a *Volume Security Group* in the *Volume Security Groups* sub-section of the *Settings* section and choose the *Key Pairs* for it. Then in the volume definition, associate the volume with the relevant *Volume Security Groups*.

! The default or built-in *Volume Provisioning Groups* do not refer to any *Volume Security Groups*. To use *Volume Provisioning Groups* with secure volume attachment, generate new *Volume Provisioning Groups* that refer to the appropriate *Volume Security Groups*.

## 3.13. CRC Check

---

Starting with **NVMesh** 2.0.5, it is possible to configure a volume with CRC verification enabled.

CRC verification, also known as the “CRC Check” feature, adds a CRC signature to every block. The CRC signature is generated as part of write operation and is verified on read operations. If a CRC signature does not match the data in the block during a read operation, the IO will be treated in the same way as a hardware bad sector and the data will be restored through the standard data protection mechanisms.

The performance impact of enabling the feature is as follows:

- **Mirrored Volumes:** The impact will be more significant for large IOs.
- **Erasured Coded Volumes:** There is no overhead for write operations. For read operations, the impact will be more significant for large IOs.

In the current implementation, overall performance for CRC verification and generation is dependent on the speed of the kernel’s CRC calculation. For most cases, this is directly related to the AVX capabilities of the processors running on the **Client** nodes. For a single 4k I/O, the latency overhead is typically around one microsecond. For high throughputs, there may be a more significant effect that is hardware-dependent. For example, on a system with dual CPUs of 16 cores each, the CRC check reduced performance from 40 GB/s to 30 GB/s.

\* Excelero recommends to test the overhead on specific hardware for precise overhead estimations.

! The CRC Check can be enabled only on volumes located on drives with Metadata Support.

## 3.14. 512-byte Block Size Emulation

---

**NVMesh 2.5.2** provides an option for attaching volumes and setting the kernel block size to 512b. 512b operation is achieved through emulation. 4k blocks are read and written to the drives. To ensure data consistency, read-modify-write operations required for operations on part of a 4k block are done under lock. Therefore, this functionality is currently available for volumes with data-protection, i.e. | **MeshProtect 1** (Mirrored), **MeshProtect 10** (Striped and Mirrored) and **MeshProtect 60** (Dual Parity).

To attach a volume with 512b block emulation, add the **—sub-block-io-allowed** flag to the ***nvmesh\_attach\_volumes*** command.

## 4. NVMesh Software Installation

---

These instructions assume familiarity on the part of the administrator with tasks such as installing packages, making changes to configuration files and general Linux systems administration knowledge. You will be guided through a sample installation, creation of a *logical volume* and attachment of that *volume* to a **client**.

### 4.1. Installation Overview

---

1. Prepare for installation
  - a. Validate the hardware and software requirements as detailed in this user guide and the [NVMesh Interoperability Matrix](#).
  - b. For RDMA setups, [Install the Mellanox OFED software](#) if required.
  - c. Review and configure the systems, for instance repository definitions, for the appropriate software delivery methods, see [Software Delivery](#).
  - d. For help in validation, install the `nvmesh-utils` package, run the `nvmeshdiag` utility and share the output with [Excelero Technical Support](#).
2. Set up the **Management Servers**:
  - a. Install the `nvmesh-management` package.
  - b. Configure specific **Management** options in `/etc/opt/NVMesh/management.js.conf`
  - c. Start the **Management**.
  - d. (Optional) Set up a management HA cluster. For more information refer to [Management Scalability](#).
3. Set up the **Clients** and **Targets**:
  - a. For both **Clients** and **Targets**:
    - i. Install the `nvmesh-core` package.
    - ii. Define the **Management Servers** addresses in `/etc/opt/NVMesh/nvmesh.conf`.
    - iii. Optionally, [Configure TCP/IP Support](#).
  - b. On **Targets**
    - i. (Only for RDDA setups) [Enable RDDA](#).
    - ii. If desired, exclude any NVMe drives that should not be used by **NVMesh 2.5.2**, see [Exclude Drives](#).
    - iii. Enable and start the `nvmeshtarget` service.
  - c. On **Clients**
    - i. Start the `nvmeshclient` service.
4. Define volumes:
  - a. Log in via a web browser.
  - b. If this is the first login to the **Management**, perform the following 2 steps:
    - i. Add a new administrative user.
    - ii. Log out, then log in as the new administrative user.
  - c. [Use the Drives screen to format the drives](#) before use by **NVMesh 2.5.2**.
  - d. In the **Management**'s GUI, choose "Volumes", and then "+".
  - e. Create a volume.

- f. Run the `nvmesh_attach_volumes` command on a **Client** to attach the volume.

## 4.2. Prerequisites

---

In order to install and run **NVMesh 2.5.2**, there are several prerequisites to be aware of and prepare for. These prerequisites fall into these main categories:

- [Hardware Dependencies](#)
- [Software Dependencies](#)
- [Network Connectivity](#)
- [Firewalls and Specific Ports](#)
- [NTP Time Synchronization](#)

### 4.2.1. Hardware Requirements

---

It is important to validate the hardware requirements as detailed in this user guide and the [NVMesh Interoperability Matrix](#).

#### 4.2.1.1. Memory Requirements

---

The following details memory usage of the **Client** and **Targets**.

##### **Clients**

**Clients** use a negligible amount of memory as data operations are synchronous and do not utilize a cache. **NVMesh 2.5.2 Clients** reserve 20MB per core for internal tracing.

##### **Targets**

**Targets** utilize memory proportional to the size of the NVMe devices being served. The amount of memory required also varies by the number of IO queues (IOQs) supported by the NVMe devices as well as the number of attached clients.

The number of IOQs supported by a drive can be found in its smart file. For each device managed by a **Target**, there is a file in the form `/proc/nvmeibs/smart<N>`. This file has a key=value format and includes the device's serial number and the number of queues, see the *Completion Queues* and *Submission Queues*. In all current known NVMe implementations, these values are the same. Following is an example:

```
[root@nvme1076 18:03:32 ~]# grep -e Serial -e Queue /proc/nvmeibs/smart[0-2]
/proc/nvmeibs/smart0:Serial Number=BTLJ91040FA71P0FGN
/proc/nvmeibs/smart0:Submission Queues=128
```

```
/proc/nvmeibs/smart0:Completion Queues=128
/proc/nvmeibs/smart1:Serial Number=S4C9NA0M400300
/proc/nvmeibs/smart1:Submission Queues=128
/proc/nvmeibs/smart1:Completion Queues=128
/proc/nvmeibs/smart2:Serial Number=S4C9NF0M500283
/proc/nvmeibs/smart2:Submission Queues=128
/proc/nvmeibs/smart2:Completion Queues=128
```

The **Target** reserves 4MB per node and 24MB per core, in addition to the **Client** reservation, for internal tracing.

#### Per TB of storage:

72MB per 1TB of drive space.

#### Per drive per **Client**:

12MB per drive, per **Client**, per channel.

The number of channels a **Client** has is determined by the `max_nr_channels` module param for `nvmeibc`, the **Client** kernel module, see [Module Parameters](#). On the target, the default value for the **Target** kernel module param `nvmeibs_nordda_io_req_num` is 96, which translates into 12MB. If this value is altered, the required memory will be 128KB per entry.

#### Per NVMe IOQ:

132K per IOQ. Example: 128 IOQ drive = 16.5MB.

Using all the information above, a fully utilized 3.2TB NVMe drive with 128 queues and 4 **Clients** utilizing it would require 294MB of target node RAM. A disaggregated “Storage Server” such as a 2U, 24 drive system with 24 of the above drives and 40 cores, serving 96 **Clients** with 4 channels each, would require a little over 140GB of RAM for the **Target**.

 The internal tracing reservation is expected to become a tunable.

## 4.2.1.2. NVMe Device Requirements

### Supported NVMe Devices

Devices which are not listed in the [NVMesh Interoperability Matrix](#) have not been tested by Excelero and may not function correctly.

## Drive Sector Size

**NVMesh 2.5.2** volumes protected by MeshProtect 60 require NVMe drives that support end-to-end data protection, sometimes referred to as having “long blocks” support or as supporting advanced metadata sector formats. This format has 4096 (4k) data bytes and 8 metadata bytes for a total of a 4104 byte sector size. This is also referred to as a 4K+8B sector format. The NVMe specification supports two ways of accessing such blocks, as inline metadata or via separate metadata pointers. **NVMesh 2.5.2** requires the latter, i.e. via separate metadata pointers.

Starting with **NVMesh 1.3**, NVMe drives must be formatted before use. NVMe drives that support 4K+8B sectors will be formatted in that sector size. If the *Drives* do not support 4K+8B, they will be formatted in 4K sectors. Lastly, if they do not support 4K+8B or 4K sectors, they will be formatted with 512B sectors.

*Drives* formatted with sector size 4K+8B can be used for any type of volume but are required for MeshProtect 60 volumes. Drives formatted with 4K or 512B sectors cannot be used in MeshProtect 60 volumes, but can be mixed in *Drive Classes*, volumes and *Volume Provisioning Groups* for other volume types.

To force a non-metadata 4K format for *Drives* that do support 4k+8B, contact [Excelero Technical Support](#) for instructions.

See [Drive Format Types](#) for instructions for verifying the supported sector formats on an NVMe drive.

### 4.2.1.3. NIC Requirements

---

As mentioned in the [Network Connectivity](#) section, **NVMesh 2.5.2 Clients** and **Targets** support TCP, RoCEv2 and Infiniband RDMA networking setups.

When configuring a non-RDMA setup, any ethernet NIC can be used. Ethernet NICs with speeds of 10 Gb/s or faster are recommended.

See [Configure TCP/IP Support](#) for more information.

## RDMA Configurations

### Mellanox ConnectX Adapters

Follow the instructions in [Enable RDDA](#) to enable RDDA on **Targets** nodes with Mellanox NICs.

## 4.2.2. Software Requirements

---

### SELinux

SELinux may prevent **NVMesh 2.5.2** kernel modules from loading successfully. Either disable SELinux or —

if SELinux is required — contact Excelero Support for recommendations, such as creating a rule that allows kernel modules that reside in the **NVMesh 2.5.2** directories to be inserted.

## nvmesh-management

 **NVMesh 2.5.2** management requires MongoDB 4.2 and NodeJS 12 LTS.

**NVMesh 2.5.2 management** uses MongoDB and NodeJS. The installation RPM has a dependency on `mongodb-org` and `nodejs`. MongoDB must be an active service (and started) before **management** can start. For information on installing MongoDB and NodeJS, see [Install MongoDB and NodeJS](#).

The following packages are required by the `nvmesh-management` package:

- `nodejs` v12.x (LTS)
- `mongodb-org` v4.2.x
- `python-argparse`

## nvmesh-core

### Red Hat Enterprise Linux/CentOS (rpm)

The following packages are required by the `nvmesh-core` package for Red Hat Enterprise Linux (RHEL) and CentOS:

- `ethtool`
- `util-linux`
- `smartmontools`
- `python-argparse`
- `python-devel`, for version 7 of the OSES and `python2-devel` for version 8

**NOTE:** a default unversioned `python` command must exist for **Client** services to start correctly.

Using the [alternatives](#) command to make the `python` command redirect to an installed Python 2 binary is one way to address this.

In addition, for RDMA environments, on service startup, the following are required:

- `libibverbs`
- `librdmacm`
- `libibcm`
- `libibmad`
- `libibumad`
- `libmlx5`

## Ubuntu (deb)

The following packages are required by the `nvmesh-core` package for Ubuntu:

- `ethtool`
- `util-linux`
- `smartmontools`
- `python-argparse`
- `python-dev`

**NOTE:** a default unversioned `python` command must exist for *Client* services to start correctly.

Using the [update-alternatives](#) command to make the `python` command redirect to an installed Python 2 binary is one way to address this.

In addition, for RDMA environments, on service startup, the following are required:

- `libibverbs1`
- `librdmacm1`
- `libibcm1`
- `libibmad5`
- `libibumad3`
- `libmlx5-1`

## SuSE

The following packages are required by the `nvmesh-core` package for SLES:

- `ethtool`
- `util-linux`
- `smartmontools`
- `python-argparse`
- `python-devel`

## 4.2.2.1. Operating System

---

Supported operating system versions can be found in the [NVMesh Interoperability Matrix](#).

## 4.2.2.2. Mellanox OFED

---

In many environments it is desired to install the Mellanox OFED software. Consult the Mellanox documentation for installation instructions.

## Mellanox OFED Prerequisites

If the kernel in use is supported by Mellanox OFED, some additional packages may be prerequisites. MLNX\_OFED installation will inform of them.

If the kernel in use is not supported by Mellanox OFED out of the box and the `--add-kernel-support` flag must be used, additional packages may be required.

## Mellanox OFED Installation

✿ It is recommended to install Mellanox OFED without the unnecessary SRP, iSER and iSER-target packages as they are not required for **NVMesh 2.5.2**. The exclusion of these packages also slims down the installation and makes it faster.

✿ The `--force-fw-update` option is recommended in order to upgrade or downgrade the NIC firmware to the revision recommended by Mellanox.

For RHEL / CentOS:

```
sudo ./mlnxofedinstall --without-srp --without-iser --without-isert --force-fw-update --force
```

**NOTE:** Other RHEL-compatible distributions, such as Rocky Linux or Arch Linux, may not be recognized by the install script. To overcome this, explicitly specify the distribution; e.g. for Rocky Linux 8.4 add `--distribution rhel8.4` to the command above.

For Ubuntu:

```
sudo ./mlnxofedinstall --without-isert-dkms --without-isert-dkms --without-srp-dkms --force-fw-update --force
```

Supported Mellanox OFED versions can be found in the [NVMesh Interoperability Matrix](#).

## 4.2.3. Network Connectivity

---

### Overview

**NVMesh 2.5.2** benefits from a network supporting RDMA, i.e. RoCEv2 or Infiniband.

### RDMA Networking

When RDMA connectivity is available, **NVMesh 2.5.2** nodes send data via RDMA. RDMA requires either

Ethernet NICs supporting RoCEv2 or InfiniBand adapters. The management control path will always utilize HTTPS over TCP/IP connectivity for functions such as **Clients** and **Targets** communication with **Management Servers**.

## RoCEv2

For Ethernet RoCEv2, reliable UDP/IP connectivity is required to establish connectivity between **Clients** and **Targets**. To this extent, it may be necessary to enable some form of flow control in the switching infrastructure with methods such as Global Pause or Priority Flow Control (PFC) or to deploy Explicit Congestion Notification (ECN) and Lossy RoCE configurations or employ Zero-touch-RoCE or combinations of these techniques. This is covered in the [RoCEv2 section](#).

## InfiniBand

For InfiniBand network fabrics, there is typically little required configuration. The only considerations for **NVMesh 2.5.2** are potentially limiting which interfaces are to be used by **NVMesh 2.5.2** and the rate at which it may query the subnet manager. In a pure InfiniBand environment or for performance considerations, it may be necessary or recommended to configure IPoIB as a communication means between the **Clients** and **Targets** to **Management Servers**, which is done over TCP/IP.

For RoCEv2 or InfiniBand, control path and data path communication may share a network connection, but may also utilize separate networks if desired.

 When employing separate Infiniband networks, each should have different subnet designations.

## Non-RDMA Networking – TCP/IP

**NVMesh 2.5.2** nodes can send data via emulated RDMA, specifically SoftiWarp. This is supported with standard Ethernet NICs.

 IPv6 is not supported in non-RDMA, standard TCP/IP configurations.

 Mixed clusters with **NVMesh 2.5.2 Clients** and **Targets** configured with both Ethernet, RoCEv2 or TCP, and Infiniband are not currently supported. For Ethernet environments, it is possible to have some inter-node communication leverage RDMA with others resorting to TCP.

## 4.2.4. Firewalls and Specific Ports

---

In general, it is simplest to install **NVMesh 2.5.2** with Linux firewalls disabled.

### Communication Ports

#### Management Nodes

✿ TCP ports 4000-4006 need to be open by default on **Management Servers**. This can be changed through configuration. See [Management Configuration](#).

By default, **Management Servers** listen on TCP ports 4000 & 4001 for administration and provisioning communication with **Clients** and ports 4002-4006 for statistics communications. These ports should be made available to them if a firewall is deployed on the server running the **Management**.

For systems using `iptables`, port 4000 can be enabled with this command (as root):

```
iptables -I INPUT 1 -m state --state NEW -m tcp -p tcp --dport 4000 -j ACCEPT
-m comment --comment Excelero-Management
```

For systems using `firewalld`, port 4000 can be enabled with this command (as root):

```
firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT 0 -p tcp --dport
4000 -j ACCEPT -m comment --comment Excelero-Management
```

Replace the “4000” in the examples above with “4001” or any other port numbers that need to be open. These ports only need to be opened on systems running **Management**.

When deploying **Management Servers** with load balancers, additional firewall configuration changes may be required.

#### Target Nodes

##### RoCEv2

✿ UDP port 4791 should be open on **Target** nodes for proper function.

For Ethernet RoCEv2 fabrics, firewalls on the **Targets** should be configured to allow traffic to UDP port 4791. This is the reserved RoCEv2 port number.

## TCP

\* UDP port 4100 and TCP ports 7914-7978 should be open on **Target** nodes for proper function.

When using TCP instead of RoCEv2 on Ethernet links for the data path, TCP ports 7914-7978 should be accessible. This is expected to be made configurable in an upcoming version. In such setups, UDP port 4100 should be accessible for TOMA communication.

## 4.2.5. NTP Time Synchronization

---

Proper time synchronization between all **NVMesh 2.5.2** cluster nodes is important for the accuracy of performance statistics and logs. It is highly recommended to configure NTP time synchronization on all servers running **NVMesh 2.5.2 Clients, Targets** and **Management Servers**.

The time synchronization between **Clients** and **Targets** with the **Management Servers** is especially required for proper recording of statistics information.

\* Consult the NTP documentation for time synchronization best practices. In general, it is recommended to configure a minimum of three time servers as primary time servers.

## 4.3. Network Configuration

---

The following section describes the steps required for setting up the network for **NVMesh 2.5.2**.

### 4.3.1. RoCEv2

---

RDMA over Converged Ethernet (RoCE) is a network protocol enabling RDMA over ethernet. RoCEv2 uses IP packets that can be routed in contrast to RoCEv1.

Communication between **NVMesh 2.5.2 Clients** and **Targets** can be run using RoCE networks.

The following sections present the main options for deploying RoCE efficiently.

1. By deploying a lossless network configuration:
  - a. Using Global Pause
  - b. Using a combination of PFC and ECN
2. By deploying a more conventional lossy network configuration:
  - a. Using Lossy RoCE

## 4.3.1.1. Global Pause

---

Global Pause is a flow control mechanism used for RoCE. It treats all network traffic, and not just RoCE traffic, in the same manner, attempting to ensure lossless behavior when configured globally. It is mainly intended for simple configuration of networks dedicated to storage or RoCE traffic.

On servers, configuring Global Pause typically comprises setting flow control pause parameters to be on for the relevant NICs. Checking their state is done with `ethtool -a <NICNAME>` and turning them on requires `ethtool -A <NICNAME> rx on tx on`.

On switches, configuring Global Pause typically requires employing commands similar to `flowcontrol send on` and `flowcontrol receive on` for all switch ports through which the relevant traffic may flow.

Note that not all switches support this functionality.

## 4.3.1.2. PFC/ECN

---

This section covers the necessary steps to successfully deploy Priority Flow Control (PFC) & Explicit Congestion Notification (ECN) together in a Mellanox Spectrum Lossless RoCE environment where hosts are connected using ConnectX Ethernet adapters.

This solution is best deployed in production environments where performance is critical.

There are two options for QoS trust: L2 (PCP) & L3 (DSCP). This section uses L3 trust which leverages the DSCP field in the IP header and is maintained across router boundaries making it ideal for use with RoCEv2 (Routed RoCE).

Minimum requirements for this configuration are as follows:

- RHEL or CentOS – 7.3 or later
- Mellanox OFED 4.1 or later
- Mellanox Spectrum Switches with MLNX-OS 3.6.5000 or later

## Example Mellanox Switch Configuration for MLNX-OS

The following steps configure ports Ethernet 1/1 through 1/12 and port-channel 1 comprised of Ethernet 1/13-1/16.

These steps assume the reader knows how to use MLNX-OS and transition in and out of configuration mode. Don't forget to save your configuration with "write memory" once you're finished. These commands may be duplicated on a second switch as needed and depending on the fabric topology.

1. **Enable ECN on TC3** – For all host-ports and inter-switch links:

```
[standalone: master] # config terminal
[standalone: master] (config) # interface ethernet 1/1-1/12 traffic-class 3 congestion-control ecn minimum-absolute 150 maximum-absolute 1500
[standalone: master] (config) # interface port-channel 1 traffic-class 3 congestion-control ecn minimum-absolute 150 maximum-absolute 1500
```

- 2. Lossless-RoCE Buffer Pool** – Define lossless buffer pool using the new buffer pool commands in 3.6.5000. A higher percentage reservation may be used if there is little to no lossy traffic expected on the fabric:

```
[standalone: master] (config) # traffic pool roce type lossless
[standalone: master] (config) # traffic pool roce memory percent 50.00
[standalone: master] (config) # traffic pool roce map switch-priority 3
```

- 3. Set CNP traffic priority** – Set strict ETS priority for CNP:

```
[standalone: master] (config) # interface ethernet 1/1-1/12 traffic-class 6 dcb ets strict
[standalone: master] (config) # interface port-channel 1 traffic-class 6 dcb ets strict
```

- 4. Set L3 QoS Trust Mode (DSCP)** – Configure the QoS trust mode:

```
[standalone: master] (config) # interface ethernet 1/1-1/12 qos trust L3
[standalone: master] (config) # interface port-channel 1 qos trust L3
```

- 5. Enable PFC** – Enable PFC on Priority 3 on all host and inter-switch link ports:

```
[standalone: master] (config) # dcb priority-flow-control enable force
[standalone: master] (config) # interface ethernet 1/1-1/12 dcb priority-flow-control mode on force
[standalone: master] (config) # interface port-channel 1 dcb priority-flow-control mode on force
```

- 6. Enable DCBX via LLDP** – Enable LLDP for host port inheritance of PFC settings using DCBX:

```
[standalone: master] (config) # lldp
```

✿ Alternatively, Mellanox has added a simpler 'one-command' method for achieving the same configuration shown above on newer versions of OnyxOS.

Minimum requirements for the one-command RoCE Lossless configuration are as follows:

- RHEL or CentOS – 7.3 or later
- Mellanox OFED 4.6 or later
- Mellanox Spectrum Switches with MLNX-OS 3.8.2008 or later

```
[standalone: master] # config terminal
[standalone: master] (config) # roce lossless
```

You can confirm the settings and their details using the following command (only where the one-command method has been used):

```
[standalone: master] # show roce

RoCE mode : lossless
LLDP : enabled
Port trust mode: L3

Application TLV:
  Selector: udp
  Protocol: 4791
  Priority: 3

Port congestion-control:
  Mode: ecn, absolute
  Min : 150
  Max : 1500

PFC : enabled
switch-priority 3: enabled

RoCE used TCs:
  _____
  Switch-Priority TC Application ETS
  _____
  3 3 RoCE WRR 50%
  6 6 CNP Strict

RoCE buffer pools:
```

```

Traffic Type Memory Switch Memory actual Usage Max Usage
Pool [%] Priorities
-----
lossy-default lossy auto 0, 1, 2, 4, 5, 6, 7 6.7M 0 1.5K
roce-reserved lossless auto 3 6.7M 0 8.9K

```

## Cumulus Linux Switch Configuration

Instructions for configuring lossless RoCE including ECN and PFC on Cumulus Linux 3.6 and Mellanox switches specifically, but with wider applicability can be found at this [link](#). This example provides specific numbers for the parameters also described below. The optimal numbers may vary with port parameters such as speed.

For newer Cumulus Linux versions, see [RoCE on Cumulus](#) for a simpler way form of configuration.

### Enabling and Configuring PFC on Cumulus Linux 4.1

The simplest way of turning on PFC on Cumulus is to add “storage-optimized pfc” to the relevant interfaces, for example as follows:

```

cumulus@switch:~$ net add interface swp1 storage-optimized pfc
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit

```

The following alternative form of configuration may apply to other and earlier versions of Cumulus Linux as well.

PFC is disabled by default. Enable it by configuring the settings shown in the example below in the file `/etc/cumulus/datapath/traffic.conf` on the switch.

Restarting the `switchd` service may be required to implement the changes.

**Note:** the optimal values for the `xon_delta`, `xoff_size` and `port_buffer_bytes` parameters are dependent on the speed of the link.

```

# to configure priority flow control on a group of ports:
# -- assign cos value(s) to the cos list
# -- add or replace a port group names in the port group list
# -- for each port group in the list
#   -- populate the port set, e.g.
#       swp1-swp4,swp8,swp50s0-swp50s3
#   -- set a PFC buffer size in bytes for each port in the group
#   -- set the xoff byte limit (buffer limit that triggers PFC frame transmit t
o start)

```

```
# -- set the xon byte delta (buffer limit that triggers PFC frame transmit to
stop)
# -- enable PFC frame transmit and/or PFC frame receive
# priority flow control
pfc.port_group_list = [pfc_port_group]
pfc.pfc_port_group.cos_list = []
pfc.pfc_port_group.port_set = swp1-swp4,swp6
pfc.pfc_port_group.port_buffer_bytes = 25000
pfc.pfc_port_group.xoff_size = 10000
pfc.pfc_port_group.xon_delta = 2000
pfc.pfc_port_group.tx_enable = true
pfc.pfc_port_group.rx_enable = true
```

See [Cumulus Link PFC](#) for more information.

## Enabling and Configuring ECN on Cumulus Linux 4.1

The simplest way of turning on ECN on Cumulus is to add “storage-optimized” to the relevant interfaces, for example as follows:

```
cumulus@switch:~$ net add interface swp1 storage-optimized
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

The following alternative form of configuration may apply to other versions of Cumulus Linux.

ECN is configured through the settings shown in the example below in the file `/etc/cumulus/datapath/traffic.conf` on the switch.

Restarting the `switchd` service may be required to implement the changes.

**Note:** the optimal values for the `min_threshold_bytes` and `max_threshold_bytes` parameters are dependent on the speed of the link. If employing PFC on the same link, make sure to set ECN to be put into use before PFC pauses the traffic using thresholds lower than `xoff_size`.

```
# Explicit Congestion Notification
# to configure ECN on a group of ports:
# -- add or replace port group names in the port group list
# -- assign cos value(s) to the cos list *ECN will only be applied to traffic ma
tching this COS*
# -- for each port group in the list
# -- populate the port set, e.g.
# swp1-swp4,swp8,swp50s0-swp50s3
ecn.port_group_list = [ecn_port_group]
ecn.ecn_port_group.cos_list = [0]
```

```
ecn.ecn_port_group.port_set = swp1-swp4,swp6
ecn.ecn_port_group.min_threshold_bytes = 40000
ecn.ecn_port_group.max_threshold_bytes = 200000
ecn.ecn_port_group.probability = 100
```

See [Cumulus Link Congestion Notification](#) for more information.

## Mellanox ConnectX Configuration

This section shows a single-path host with only one port used for **NVMesh 2.5.2** (ens3). These steps may be repeated for each additional interface on multipath hosts. Configuring dual-port adapters requires the use of either the ‘\_P1’ or ‘\_P2’ options in the command example below as each port’s settings are independent of one another. If both are used for **NVMesh 2.5.2**, then both must have the same settings applied.

### Enable PFC & ECN on Priority 3

Using MST to set adapter firmware settings (repeat the *mlxconfig* command once for each adapter port [P1 or P2] to be used as an **NVMesh 2.5.2** interface). The `/dev/mst/mt####_pciconf#` device will vary from system to system and depends on which model and the quantity of adapters installed. The `mst status` command may be used to list them all and identify the ones needing modification.

**! The following changes are non-volatile. A reboot is required to commit adapter firmware changes.**

```
# mst start

Starting MST (Mellanox Software Tools) driver set
Loading MST PCI module - Success
Loading MST PCI configuration module - Success
Create devices
-W- Missing lsusb command, skipping MTUSB devices detection
Unloading MST PCI module (unused) - Success

# mlxconfig -d /dev/mst/mt4119_pciconf0 -y set \
ROCE_CC_PRIO_MASK_P1=8 \
RPG_THRESHOLD_P1=1 \
DCE_TCP_G_P1=1019 \
LLDP_NB_DCBX_P1=TRUE \
LLDP_NB_TX_MODE_P1=2 \
LLDP_NB_RX_MODE_P1=2
```

```

Device #1:
-----

Device type:    ConnectX5
PCI device:    /dev/mst/mt4119_pciconf0

Configurations:
                Next Boot      New
ROCE_CC_PRIO_MASK_P1      0          8
RPG_THRESHOLD_P1          5          1
DCE_TCP_G_P1              4         1019
LLDP_NB_DCBX_P1           False(0)   True(1)
LLDP_NB_TX_MODE_P1        OFF(0)     ALL(2)
LLDP_NB_RX_MODE_P1        OFF(0)     ALL(2)

Apply new Configuration? ? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.

```

## Create a custom /sbin/ifup-local script

Create the `/sbin/ifup-local` file using the Linux network interface name and IB device name (`mlx5_#`) for each port which needs to be configured. The below example uses `ens3` and `mlx5_0` with no other interfaces defined, but could also include other interfaces in additional case statement stanzas as noted in the comment of the file.

```

#!/bin/bash
#
# Script to apply lossless-RoCE QoS settings for NVMesh interfaces
#

case "$1" in

    ens3)
        echo "Configuring DCBX PFC control, QoS L3-trust (DSCP) and RoCE ToS"
        /usr/bin/mlnx_qos -i ens3 -d fw --trust dscp
        /usr/sbin/cma_roce_tos -d mlx5_0 -t 106
        ;;

# repeat <ifname>) stanzas for each interface used by NVMesh

esac
exit 0

```

Set the file to mode 755:

```
# chmod 755 /sbin/ifup-local
```

 Reboot the host to commit all of the above changes.

## Validate PFC/ECN Setup

### Confirm QoS Trust & PFC settings

To confirm PFC and QoS Trust adapter firmware settings applied after reboot (repeat for each **NVMesh 2.5.2** interface):

```
# mst start

Starting MST (Mellanox Software Tools) driver set
Loading MST PCI module - Success
Loading MST PCI configuration module - Success
Create devices
-W- Missing lsusb command, skipping MTUSB devices detection
Unloading MST PCI module (unused) - Success

# mlxconfig -d /dev/mst/mt4119_pciconf0 query | egrep -e ROCE_CC_PRIO_MASK -e RP
G_THRESHOLD -e DCE_TCP_G -e LLDP

          ROCE_CC_PRIO_MASK_P1           8
          RPG_THRESHOLD_P1               1
          DCE_TCP_G_P1                   1019
          LLDP_NB_DCBX_P1                 True(1)
          LLDP_NB_RX_MODE_P1              ALL(2)
          LLDP_NB_TX_MODE_P1              ALL(2)

# mlnx_qos -i ens3

DCBX mode: Firmware controlled
Priority trust state: dscp
dscp2prio mapping:
  prio:0 dscp:07,06,05,04,03,02,01,00,
  prio:1 dscp:15,14,13,12,11,10,09,08,
  prio:2 dscp:23,22,21,20,19,18,17,16,
  prio:3 dscp:31,30,29,28,27,26,25,24,
```

```

prio:4 dscp:39,38,37,36,35,34,33,32,
prio:5 dscp:47,46,45,44,43,42,41,40,
prio:6 dscp:55,54,53,52,51,50,49,48,
prio:7 dscp:63,62,61,60,59,58,57,56,
Receive buffer size (bytes): 262016,262016,0,0,0,0,0,0,0,
Cable len: 7
PFC configuration:
    priority    0    1    2    3    4    5    6    7
    enabled     0    0    0    1    0    0    0    0
    buffer      0    0    0    1    0    0    0    0
tc: 0 ratelimit: unlimited, tsa: ets, bw: 50%
    priority: 0
    priority: 1
    priority: 2
    priority: 4
    priority: 5
    priority: 7
tc: 3 ratelimit: unlimited, tsa: ets, bw: 50%
    priority: 3
tc: 6 ratelimit: unlimited, tsa: vendor
    priority: 6

```

## Verify that Global Pause is no longer enabled

Verify that Global Pause (Rx/Tx Flow control) is no longer enabled on the Linux network interface:

```
# ethtool -a ens3
```

Pause parameters for ens3:

```
Autonegotiate:      off
RX:                 off
TX:                 off
```

## Validate interface throughput performance

Use the `ib_send_bw` tool to validate the throughput is as expected per the configured link rates:

```
Server: # ib_send_bw -d mlx5_0 -R -F -D5 --report_gbits
```

```
Client: # ib_send_bw -d mlx5_0 -R -F -D5 --report_gbits <server_IP>
```

## Monitor PFC/ECN Setup

In this section, you will find some important counters you can watch to verify that PFC pause traffic and ECN throttling is happening in your fabric.

### Important OS Counters to Monitor PFC/ECN traffic flow

#### Infiniband ECN/CNP counters

Mellanox Hardware Counters (by mlx\_# device; run this loop for each device port)

```
# watch -n 1 'for i in `ls /sys/class/infiniband/mlx5_0/ports/1/hw_counters|egrep -e cnp -e ecn`;do echo $i;cat /sys/class/infiniband/mlx5_0/ports/1/hw_counters/$i;done'
```

#### Linux interface traffic counters by priority

Display traffic counters for prio0 (TCP), prio3 (RoCE), and prio6 (CNP):

```
# watch -n 1 'ethtool -S ens3 | egrep -e prio0 -e prio3 -e prio6'
```

## 4.3.1.3. Lossy RoCE

---

Lossy RoCE is a recent development in the domain of RoCE intended to minimize the switch configuration required to run RoCE successfully with little sacrifice in performance.

For more details, see [Enable & Disable Lossy RoCE](#).

To persist the settings, ensure the commands required to enable Lossy RoCE are issued each time NIC ports are brought up. There are various means for doing this dependent on operating system flavor and mode of networking configuration, i.e. with or without NetworkManager. `/sbin/ifup-local` is often used for this purpose.

**NVMesh 2.5.2** has a beta-level option for persisting the settings. Contact [Excelero Technical Support](#) for instructions.

## 4.3.2. RoCEv2 Multipathing

---

This section describes the best practices for setting up RoCE interfaces to support highly available multipath deployments. This method uses `iproute2` to create succinct route tables and policies for each RoCE interface; known as Policy-based Routing. This is done to ensure that every RoCE interface in the **NVMesh 2.5.2** fabric may communicate regardless of their connected VLAN and IP differences. This configuration

also supports locking with RDMA atomics as used by **NVMesh 2.5.2** for distributed lock control and communication.

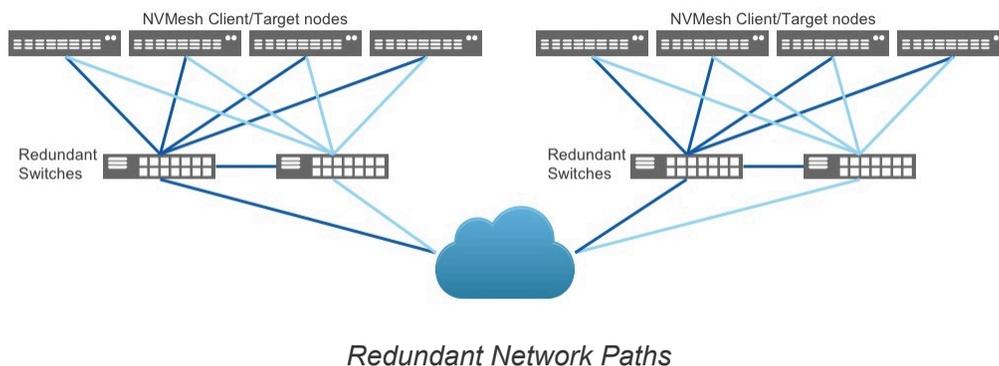
Bonding and LAG configurations are currently not supported by **NVMesh 2.5.2**.

**!** For optimized lock recovery in a failure scenario, all adapters used on disaggregated targets or converged nodes should be dual-ported with both ports cabled to different switches and in different subnets. This is because RDMA atomic locks are global to an adapter and not the whole of the server. An upcoming implementation will relieve this limitation.

## 4.3.2.1. Dual Switch Requirements

For network resilience, it is recommended to deploy more than a single switch. To connect the **NVMesh 2.5.2** cluster to dual switches, follow these guidelines:

- All network interfaces must be redundantly cabled as shown in the topology diagram in the following figure:



- The switches should be interconnected, either through a pair of spine switches (not shown), or through cross-connect cabling (shown above). If cross-connecting, be careful to include enough links to support an acceptable ratio of client port bandwidth to interconnect bandwidth. This has the potential of becoming a bottleneck in failure scenarios if insufficient bandwidth is available between switches or leaf to spine.
- Every RoCE port on a single server must be in a unique subnet, which translates to a unique VLAN in the RoCE fabric. In the diagram above, we have 2 VLANs shown in Dark Blue and Light Blue. The **Targets** and **Clients** are connected to both VLANs and are connected across both switches to ensure redundancy.
- In this design, both switches in the dual-switch topology have unique VLANs. For example, Switch 1 is configured for VLAN 100 and Switch 2 is configured for VLAN 200.
- The port-channel used to interconnect the switch pair must be in LAG, **not MLAG mode**.
- The port-channel must be configured as a router port (L3), not switch port (L2).
- Routing between the different subnets across both switches is accomplished with static routes

pointing to the far end router port as the gateway to the SVIs (VLAN router IP's) for the remote subnets.

- For additional performance, switch interfaces should be configured for Jumbo Frames at 9216 bytes. Hosts will use a smaller size of 4200 as recommended by Mellanox for RoCE. This has been tested and shown improvements in performance for certain use-cases.
- Some form of flow control (Global Pause, PFC, PFC+ECN or Lossy RoCE) should be implemented. This document demonstrates Global Pause for the sake of simplicity.
- For Infiniband networks, there is no need for the VLAN configuration,

Another alternative is to use a single LAN or VLAN for all NIC ports. In this case, it is important to avoid ARP issues by setting `arp_ignore` to 1 and `arp_announce` to 2 for these interfaces in the kernel. One way of doing this would be to embed this lines into a file in `/etc/sysctl.d`, for instance `/etc/sysctl.d/90-lan-multipath.conf`, which would set this definition for all NICs in the server.

```
net.ipv4.conf.all.arp_ignore=1
net.ipv4.conf.all.arp_announce=2
```

See [ip-sysctl](#) for more information on these system configurations.

✿ After making changes to these parameters, it is recommended to flush ARP caches, using `ip -s -s neigh flush all` and then verify that the network is working as expected using pings and checking the arp caches using `ip neigh show`.

## 4.3.2.2. Linux OS Requirements

- Network interface addresses and settings can be set using various methods including configuration files, NetworkManager, or netplan. The exact tools used depend on your Linux distribution and preferences. The detailed example of configuring policy-based routing that is outlined in [the section below](#) uses configuration files, but it is possible to use NetworkManager configuration profiles or netplan to configure similar routing tables, and to employ route metrics to establish the relative priorities of these routes.
- Reverse Path Filtering must be set to permissive mode on all RoCE interfaces. (`net.ipv4.conf.<interface>.rp_filter=2`).
- ARP settings for the RoCE ports will require some tuning to change the default Linux OS behavior. Basically, a multi-homed Linux server sees all IP addresses as belonging to the whole of the system and not as bound to individual interface ports or functions. The setting changes documented below ensure the OS understands that IPs are associated to individual ports and not just the whole of the system.
- Disable automatic private IP addressing, also known as zeroconf: `echo "NOZEROCONF=yes" >> /etc/sysconfig/network.`

## 4.3.2.3. Switch Configuration Examples

### MLNX-OS / ONYX

\* Do not use the Global Pause `flowcontrol` settings for ports as shown below if you plan to implement PFC+ECN. Global Pause is not supported in combination with PFC+ECN.

#### Mellanox Spectrum (SN2100)

##### Switch1

```
##
## Interface Ethernet configuration
##
interface port-channel 1
interface ethernet 1/1-1/6 mtu 9216 force
interface ethernet 1/7/1-1/7/4 mtu 9216 force
interface ethernet 1/9/1-1/9/4 mtu 9216 force
interface ethernet 1/11-1/12 mtu 9216 force
interface ethernet 1/13/1-1/13/4 mtu 9216 force
interface ethernet 1/15-1/16 mtu 9216 force
interface port-channel 1 mtu 9216 force
interface ethernet 1/1-1/6 flowcontrol receive on force
interface ethernet 1/1-1/6 flowcontrol send on force
interface ethernet 1/7/1-1/7/4 flowcontrol receive on force
interface ethernet 1/7/1-1/7/4 flowcontrol send on force
interface ethernet 1/9/1-1/9/4 flowcontrol receive on force
interface ethernet 1/9/1-1/9/4 flowcontrol send on force
interface ethernet 1/11-1/12 flowcontrol receive on force
interface ethernet 1/11-1/12 flowcontrol send on force
interface ethernet 1/13/1-1/13/4 flowcontrol receive on force
interface ethernet 1/13/1-1/13/4 flowcontrol send on force
interface ethernet 1/15-1/16 flowcontrol receive on force
interface ethernet 1/15-1/16 flowcontrol send on force
interface port-channel 1 flowcontrol receive on force
interface port-channel 1 flowcontrol send on force
interface ethernet 1/11-1/12 channel-group 1 mode active
interface ethernet 1/15-1/16 channel-group 1 mode active

##
## LAG configuration
```

```
##
lacp
port-channel load-balance ethernet source-destination-ip source-destination-ma
c

##
## VLAN configuration
##
vlan 100
interface ethernet 1/1 switchport access vlan 100
interface ethernet 1/3 switchport access vlan 100
interface ethernet 1/5 switchport access vlan 100
interface ethernet 1/7/1 switchport access vlan 100
interface ethernet 1/7/3 switchport access vlan 100
interface ethernet 1/9/1 switchport access vlan 100
interface ethernet 1/9/3 switchport access vlan 100
interface ethernet 1/13/1 switchport access vlan 100
interface ethernet 1/13/3 switchport access vlan 100
vlan 150
interface ethernet 1/2 switchport access vlan 150
interface ethernet 1/4 switchport access vlan 150
interface ethernet 1/6 switchport access vlan 150
interface ethernet 1/7/2 switchport access vlan 150
interface ethernet 1/7/4 switchport access vlan 150
interface ethernet 1/9/2 switchport access vlan 150
interface ethernet 1/9/4 switchport access vlan 150
interface ethernet 1/13/2 switchport access vlan 150
interface ethernet 1/13/4 switchport access vlan 150
vlan 100 name "VLAN 100"
vlan 150 name "VLAN 150"
mac-address-table aging-time 1800

##
## STP configuration
##
no spanning-tree

##
## L3 configuration
##
ip routing vrf default
interface port-channel 1 no switchport force
interface vlan 100
interface vlan 150
```

```
interface port-channel 1 ip address 10.0.0.1 255.255.255.252
interface vlan 100 counters
interface vlan 100 ip address 192.168.100.1 255.255.255.0
interface vlan 100 mtu 9216
interface vlan 150 counters
interface vlan 150 ip address 192.168.150.1 255.255.255.0
interface vlan 150 mtu 9216
ip route 192.168.200.0 /24 10.0.0.2
ip route 192.168.250.0 /24 10.0.0.2

##
## LLDP configuration
##
    lldp
<...truncated...>
```

## Switch2

```
##
## Interface Ethernet configuration
##
interface port-channel 1
interface ethernet 1/1-1/6 mtu 9216 force
interface ethernet 1/7/1-1/7/4 mtu 9216 force
interface ethernet 1/9/1-1/9/4 mtu 9216 force
interface ethernet 1/11-1/12 mtu 9216 force
interface ethernet 1/13/1-1/13/4 mtu 9216 force
interface ethernet 1/15-1/16 mtu 9216 force
interface port-channel 1 mtu 9216 force
interface ethernet 1/1-1/6 flowcontrol receive on force
interface ethernet 1/1-1/6 flowcontrol send on force
interface ethernet 1/7/1-1/7/4 flowcontrol receive on force
interface ethernet 1/7/1-1/7/4 flowcontrol send on force
interface ethernet 1/9/1-1/9/4 flowcontrol receive on force
interface ethernet 1/9/1-1/9/4 flowcontrol send on force
interface ethernet 1/11-1/12 flowcontrol receive on force
interface ethernet 1/11-1/12 flowcontrol send on force
interface ethernet 1/13/1-1/13/4 flowcontrol receive on force
interface ethernet 1/13/1-1/13/4 flowcontrol send on force
interface ethernet 1/15-1/16 flowcontrol receive on force
interface ethernet 1/15-1/16 flowcontrol send on force
interface port-channel 1 flowcontrol receive on force
interface port-channel 1 flowcontrol send on force
```

```
interface ethernet 1/11-1/12 channel-group 1 mode active
interface ethernet 1/15-1/16 channel-group 1 mode active

##
## LAG configuration
##
lacp
port-channel load-balance ethernet source-destination-ip source-destination-ma
c

##
## VLAN configuration
##
vlan 200
interface ethernet 1/1 switchport access vlan 200
interface ethernet 1/3 switchport access vlan 200
interface ethernet 1/5 switchport access vlan 200
interface ethernet 1/7/1 switchport access vlan 200
interface ethernet 1/7/3 switchport access vlan 200
interface ethernet 1/9/1 switchport access vlan 200
interface ethernet 1/9/3 switchport access vlan 200
interface ethernet 1/13/1 switchport access vlan 200
interface ethernet 1/13/3 switchport access vlan 200
vlan 250
interface ethernet 1/2 switchport access vlan 250
interface ethernet 1/4 switchport access vlan 250
interface ethernet 1/6 switchport access vlan 250
interface ethernet 1/7/2 switchport access vlan 250
interface ethernet 1/7/4 switchport access vlan 250
interface ethernet 1/9/2 switchport access vlan 250
interface ethernet 1/9/4 switchport access vlan 250
interface ethernet 1/13/2 switchport access vlan 250
interface ethernet 1/13/4 switchport access vlan 250
vlan 200 name "VLAN 200"
vlan 250 name "VLAN 250"
mac-address-table aging-time 1800

##
## STP configuration
##
no spanning-tree

##
## L3 configuration
```

```
##
ip routing vrf default
interface port-channel 1 no switchport force
interface vlan 200
interface vlan 250
interface port-channel 1 ip address 10.0.0.2 255.255.255.0
interface vlan 200 counters
interface vlan 200 ip address 192.168.200.1 255.255.255.0
interface vlan 200 mtu 9216
interface vlan 250 counters
interface vlan 250 ip address 192.168.250.1 255.255.255.0
interface vlan 250 mtu 9216
ip route 192.168.100.0 /24 10.0.0.1
ip route 192.168.150.0 /24 10.0.0.1

##
## LLDP configuration
##
lldp
<...truncated...>
```

## 4.3.2.4. Linux OS Configuration Examples

Below are the ARP settings combined into a `sysctl` configuration file. Add the lines in the example below using the correct Linux network interface names into a file located here: `/etc/sysctl.d/50-roce-multipath.conf`

Persist the settings immediately using the following command: `# sysctl -f /etc/sysctl.d/50-roce-multipath.conf`

### **`/etc/sysctl.d/50-roce-multipath.conf`:**

```
# Sets the arp replies to only egress the connected interface for a given subnet
net.ipv4.conf.all.arp_ignore=2

# Sets arp notifications to occur when a device comes up
net.ipv4.conf.all.arp_notify=1

# Announcing the local source IP address from IP packets in ARP requests sent on
interface
net.ipv4.conf.all.arp_announce=1
```

```
# Set reverse path filtering to permissive mode for configured RoCE interfaces
net.ipv4.conf.enp143s0f0.rp_filter=2
net.ipv4.conf.enp143s0f1.rp_filter=2
net.ipv4.conf.ens1f0.rp_filter=2
net.ipv4.conf.ens1f1.rp_filter=2
```

- ✿ After making changes to these parameters, it is recommended to flush ARP caches, using `ip -s -s neigh flush all` and then verify that the network is working as expected using pings and checking the arp caches using `ip neigh show`.

## 4.3.2.4.1. ifcfg Files

Below are examples of the `/etc/sysconfig/network-scripts/ifcfg-*` files for a single disaggregated target system with two dual-port adapters.

- ✿ This method serves as an example, similar configurations can be built using NetworkManager or netplan.

### ifcfg-enp143s0f0

```
NM_CONTROLLED=no
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
NAME=enp143s0f0
DEVICE=enp143s0f0
IPADDR=192.168.100.11
PREFIX=24
MTU=4200
```

### ifcfg-enp143s0f1

```
NM_CONTROLLED=no
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
NAME=enp143s0f1
DEVICE=enp143s0f1
```

```
IPADDR=192.168.250.11
PREFIX=24
MTU=4200
```

## ifcfg-ens1f0

```
NM_CONTROLLED=no
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
NAME=ens1f0
DEVICE=ens1f0
IPADDR=192.168.200.11
PREFIX=24
MTU=4200
```

## ifcfg-ens1f1

```
NM_CONTROLLED=no
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
NAME=ens1f1
DEVICE=ens1f1
IPADDR=192.168.150.11
PREFIX=24
MTU=4200
```

## 4.3.2.4.2. rt\_tables File

---

The `iproute2` command set is used for TCP/IP traffic management and monitoring. The `route-*` and `rule-*` files described in the next two sections of the manual are used to define explicit routes between the subnets defined in the Ethernet Storage Fabric. In the examples in [Section 4.3.2](#), we used subnets 192.168.100.0/24, 192.168.200.0/24, 192.168.150.0/24, and 192.168.250.0/24. Hosts are configured such that each IP storage port has a clear routed path to the others, not only in the same subnet, but all ports in all storage subnets.

 If you are using NetworkManager or netplan to configure your interfaces, connections, and routes, the procedures for configuring policy-based routing between each RoCEv2 subnet

will vary from this example.

## /etc/iproute2/rt\_tables

```
#
# reserved values
#
255     local
254     main
253     default
0       unspec
#
# local
#
#1      inr.ruhep
100    roce1
200    roce2
150    roce3
250    roce4
```

## 4.3.2.4.3. route Files

Below are examples of the `/etc/sysconfig/network-scripts/route-*` files for a single disaggregated target system with two dual-port adapters:

 If you are using NetworkManager or netplan to configure your interfaces, connections, and routes, the procedures for configuring policy-based routing between each RoCEv2 subnet will vary from this example.

### route-enp143s0f0

```
192.168.100.0/24 dev enp143s0f0 table roce1
192.168.250.0/24 via 192.168.100.1 table roce1
192.168.200.0/24 via 192.168.100.1 table roce1
192.168.150.0/24 via 192.168.100.1 table roce1
```

## route-enp143s0f1

```
192.168.250.0/24 dev enp143s0f1 table roce2
192.168.100.0/24 via 192.168.250.1 table roce2
192.168.150.0/24 via 192.168.250.1 table roce2
192.168.200.0/24 via 192.168.250.1 table roce2
```

## route-ens1f0

```
192.168.200.0/24 dev ens1f0 table roce3
192.168.150.0/24 via 192.168.200.1 table roce3
192.168.100.0/24 via 192.168.200.1 table roce3
192.168.250.0/24 via 192.168.200.1 table roce3
```

## route-ens1f1

```
192.168.150.0/24 dev ens1f1 table roce4
192.168.200.0/24 via 192.168.150.1 table roce4
192.168.250.0/24 via 192.168.150.1 table roce4
192.168.100.0/24 via 192.168.150.1 table roce4
```

## 4.3.2.4.4. rule Files

---

Below are examples of the `/etc/sysconfig/network-scripts/rule-*` files for a single disaggregated target system with two dual-port adapters:

- ✿ If you are using NetworkManager or netplan to configure your interfaces, connections, and routes, the procedures for configuring policy-based routing between each RoCEv2 subnet will vary from this example.

## rule-enp143s0f0

```
from 192.168.100.10/32 table roce1 prio 301
to 192.168.100.0/24 table roce1 prio 401
to 192.168.150.0/24 table roce1 prio 501
to 192.168.200.0/24 table roce1 prio 601
to 192.168.250.0/24 table roce1 prio 701
```

## rule-enp143s0f1

```
from 192.168.250.10/32 table roce2 prio 302
to 192.168.250.0/24 table roce2 prio 402
to 192.168.100.0/24 table roce2 prio 502
to 192.168.150.0/24 table roce2 prio 602
to 192.168.200.0/24 table roce2 prio 702
```

## rule-ens1f0

```
from 192.168.200.10/32 table roce3 prio 303
to 192.168.200.0/24 table roce3 prio 403
to 192.168.100.0/24 table roce3 prio 503
to 192.168.150.0/24 table roce3 prio 603
to 192.168.250.0/24 table roce3 prio 703
```

## rule-ens1f1

```
from 192.168.150.10/32 table roce4 prio 304
to 192.168.150.0/24 table roce4 prio 404
to 192.168.100.0/24 table roce4 prio 504
to 192.168.200.0/24 table roce4 prio 604
to 192.168.250.0/24 table roce4 prio 704
```

## 4.3.3. TCP/IP Support

---

**NVMesh 2.5.2** can be configured to utilize TCP/IP Ethernet networks. This is also referred to as *standard TCP/IP support* as opposed to RoCEv2 which requires specific NICs and requires certain network configurations to work optimally.

- ✿ Standard Ethernet NICs are supported with MTU sizes of 1500 to 9000 bytes. IPv6 is not supported in non-RDMA, standard TCP/IP configurations. Mixed clusters with **NVMesh 2.5.2 Clients** or **Targets** configured with both Ethernet and Infiniband are not supported.

To enable TCP/IP connectivity, set the following in `/etc/opt/NVMesh/nvmesh.conf`: `TCP_ENABLED="Yes"`.

To disable RDMA and use only TCP/IP, use `/etc/opt/NVMesh/nvmesh.conf`: `TCP_ONLY="Yes"`.

It is possible to use both TCP and RoCE to access a single **Target** via the same NIC. Some **Clients** may access the **Target** via TCP while others do so via RoCE. A **Client** will only use one of these communication methods to access **Targets**. The **Client** and **Target** will prefer the RoCE option. If the NICs on the **Client** and **Target** are identified as being RoCE capable and the TCP\_ONLY setting has not been set, RoCE will be used. There is no automatic fallback in case of failure to TCP.

! For a **Target** to offer both TCP and RDMA, it **MUST** use the inbox RDMA driver, and **NOT** OFED.

## 4.4. Software Delivery

---

### NVMesh 2.5.2 Packages

**NVMesh 2.5.2** software components for Linux are generally delivered in the form of RPM or DEB packages, for example:

For instance, for Red Hat Enterprise Linux 7.9:

```
nvmesh-core-2.4.0-641.el7_9.x86_64.rpm
nvmesh-utils-2.4.0-154.el7_9.x86_64.rpm
nvmesh-management-2.4.0-154.el7_9.x86_64.rpm
```

Alternatively, for Ubuntu 20.04:

```
nvmesh-core-2.4.0-641.ubuntu20_04_amd.deb
nvmesh-utils-2.4.0-154.ubuntu20_04_amd.deb
nvmesh-management-2.4.0-154.ubuntu20_04_amd.deb
```

The **nvmesh-management** and **nvmesh-core** packages, which are dependent on **nvmesh-utils**, can be installed together or independently.

\* An account and password should have been made available to you or you can request access to the repository by sending email to [support@excelero.com](mailto:support@excelero.com). For a list of Linux distributions, kernels and OFED versions supported refer to the [NVMesh Interoperability Matrix](#).

### 4.4.1. Software Delivery Red Hat / CentOS

---

The packages can be installed via any valid RPM installation method. `yum` can be used to install and

manage the packages. Excelero offers an authenticated internet accessible `yum` repository at <https://repo.excelero.com/nvmesh/2.5.2/redhat/>

If you will be using `yum` with remote repositories, once you have an account and password, create a `yum` repository configuration file: `/etc/yum.repos.d/nvmesh.repo` with contents like the following example:

```
[NVMesh]
name=NVMesh repository
baseurl=https://[user]:[password]@repo.excelero.com/nvmesh/2.5.2/redhat/[version]/x86_64/
gpgcheck=0
enabled=1
```

You should replace `user` and `password` in the above example with those supplied by Excelero and `version` to match your OS distribution version.

For example, for Red Hat Enterprise Linux v7.8 with user **jake** and password **moonwalker**:

```
[NVMesh]
name=NVMesh repository
baseurl=https://jake:moonwalker@repo.excelero.com/nvmesh/2.5.2/redhat/7.8/x86_64/
gpgcheck=0
enabled=1
```

## 4.4.2. Software Delivery Ubuntu

---

The packages can be installed using the `apt` installation method. `apt` can be used to install and manage the packages. Excelero offers an authenticated internet accessible `apt` repository at <https://repo.excelero.com/nvmesh/2.5.2/ubuntu/>

If you will be using `apt` with remote repositories, once you have an account and password, add the **NVMesh 2.5.2** repository to the local `apt` client repository list, as follows.

Add your username and password by adding the following lines to `/etc/apt/auth.conf`.

```
machine repo.excelero.com
login <USER>
password <PASSWORD>
```

Generate a file named `/etc/apt/sources.list.d/nvmesh.list` that will point to the appropriate location within the repo and obtain the Excelero GPG key.

For Ubuntu 20.04 (focal), use:

```
echo 'deb [arch=amd64] https://repo.excelero.com/repos/nvmesh/2.5.2/ubuntu/20.04/x86_64 focal main' | sudo tee /etc/apt/sources.list.d/nvmesh.list
wget -O - https://<USER>:<PASSWORD>@repo.excelero.com/repos/nvmesh/2.5.2/ubuntu/20.04/x86_64/conf/nvmesh.gpg.key | sudo apt-key add -
```

Now, fetch the latest changes from the NVMesh repository specifically:

```
sudo apt -o Dir::Etc::sourcelist="sources.list.d/nvmesh.list" -o Dir::Etc::sourceparts="-" -o APT::Get::List-Cleanup="0" update
```

Check that the package is available after fetching the changes:

```
sudo apt list | grep nvmesh
```

To install the nvmesh-core package for example, use:

```
sudo apt install nvmesh-core
```

## 4.5. Installation Instructions

---

The following sections provide detailed instructions for the installation and configuration of **NVMesh 2.5.2**.

For an overview, see [Installation Overview](#).

### 4.5.1. Install MongoDB and NodeJS

---

**Management Servers** use MongoDB as a persistent data store and NodeJS for WebUI and API services. If your Linux distribution does not include MongoDB or NodeJS packages, it will be necessary to add them using Internet hosted repos. Alternatively, you may download the packages manually if Internet access is not available.

 The following instructions assume an RPM-based Linux distribution, such as Red Hat Enterprise Linux or CentOS, with root access. Adjustments for use with sudo, or other privilege escalation tools, may be required.

# Install MongoDB

 **NVMesh 2.5.2** requires MongoDB 4.2.

**Note:** Previous versions of **NVMesh** required MongoDB 3.6. If upgrading to **NVMesh 2.5.2**, upgrading MongoDB is also required. First, upgrade to Mongo 4.0 and then continue to upgrade to 4.2. See [Upgrading](#).

Detailed instructions for installing MongoDB 4.2 can be found [here for Redhat/CentOS](#) and [here for Ubuntu](#)

For Redhat/CentOS, create `/etc/yum.repos.d/mongodb-org-4.2.repo` with the following contents:

```
[mongodb-org-4.2]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.2/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.2.asc
```

Then run `yum install -y mongodb-org` unless a specific subversion is needed.

For Ubuntu, perform the following steps:

- `wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -` to import Mongo's public key
- `echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list` to create a file list for Mongo
- `sudo apt-get update` to reload the package file list
- `sudo apt-get install -y mongodb-org` to install Mongo itself unless a specific subversion is needed.

After installation, check whether the service started cleanly and verify it is listening on `tcp/27017`:

```
root@n1101:/root# systemctl status mongod
* mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor prese
t: enabled)
   Active: active (running) since Sun 2020-07-12 14:45:38 IDT; 2 days ago
     Docs: https://docs.mongodb.org/manual
  Main PID: 2996 (mongod)
    Memory: 25.2G
```

```
CGroup: /system.slice/mongod.service
        \-2996 /usr/bin/mongod --config /etc/mongod.conf

Jul 12 14:45:38 n1101 systemd[1]: Started MongoDB Database Server.
```

```
[root@nvme141 ~]# lsof -i -nP | grep mongod
mongod 3171 mongod 6u IPv4 21985 0t0 TCP 127.0.0.1:27017 (LISTEN)
```

If you do not see `mongod` started, start it by running `systemctl start mongod`.

To enable the service to start at reboot run `systemctl enable mongod`.

 By default, `mongod` only listens on the loopback address. This is only sufficient for standalone installations of **Management**.

To install **Management** in a high availability setup, refer to [Management Scalability](#).

## Configuring Authentication for MongoDB

See [Enable Access Control for MongoDB](#) for instructions for configuration authentication to enable access control for MongoDB.

See [MongoDB Connection Options](#) for guidance on configuring authenticated access to MongoDB from the **Management Servers**.

## Install NodeJS

 **NVMesh 2.5.2** requires NodeJS 12.x LTS. Installations using previous releases using NodeJS 6.x or 8.x will need to be upgraded to NodeJS 12.x prior to installing the latest **Management** release.

To install, setup the `nodesource` repository for your Linux distribution as described [here](#). Be sure you are pointing to the `setup_12.x` script.

```
curl --silent --location https://rpm.nodesource.com/setup_12.x | sudo -E bash -
for Red Hat.
```

```
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash - for Ubuntu.
```

After preparing the repository, install NodeJS, as follows:

Operating System	Command
------------------	---------

Red Hat / CentOS	<code>sudo yum -y install nodejs</code>
Ubuntu	<code>sudo apt-get install -y nodejs</code>

**Note:** For some operating systems, such as Ubuntu 18.04, it is necessary to replace libcurl4 with libcurl3 to facilitate proper functionality of NodeJS 12.x.

## 4.5.2. Install the NVMesh Management Server

Install the **nvmesh-management** package on the server(s) you want to act as **Management**, as root:

For Redhat/CentOS: `yum -y install nvmesh-management`

For Ubuntu: `apt install -y nvmesh-management`

Upon successful installation, **Management** will be set to automatically start at boot time. Deploying for HA is covered as an advanced topic in the [Management Scalability](#) section.

After installing the package, there are **Management** specific options you might want to set in:

```
/etc/opt/NVMesh/management.js.conf
```

These include options such as setting an outbound Email server and encryption settings for web services. These options are omitted from this example. Details on this configuration file can be found in [Management Server Options](#).

Once configured, start the service:

```
systemctl start nvmeshmgr
```



If the service status is failed, you may need to reset systemd using the `systemctl reset-failed` command.

At this point, **Management** should be active.

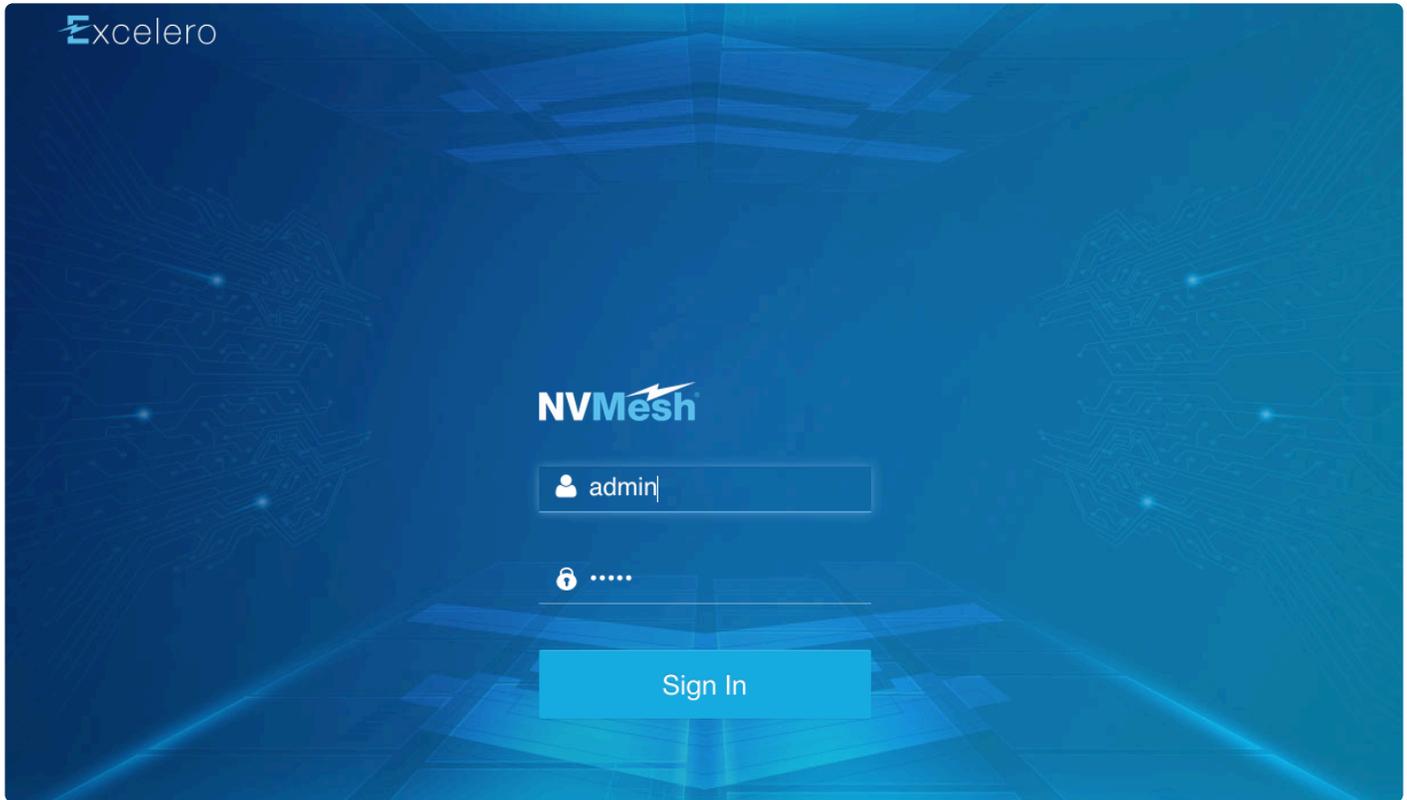
Record the IP address or hostname of the **Management Servers** as this will be required to configure the **Clients** and **Targets**. `172.10.100.201` will be using in this example.

To verify the management database was initialized properly and that **Management** is active, use a browser and attempt to connect to:

```
https://172.10.100.201:4000
```

(substitute your **Management** IP address or hostname)

If things are working properly, you should see the following login screen:



If you cannot connect to **Management**, please be sure to check if the ports are being blocked by a [firewall process](#) on **Management**.

✿ Your browser may request verification of access to a non-secure site as the website's security certificate may not be considered valid (as is it self-signed). If your organization has internal SSL certificates you can replace the default Excelero certificate with one of your own.

The default login is admin/admin. Upon initial login, you will be prompted to change the password. This is recommended, but not required.

## 4.5.2.1. Deploying NVMesh Management in Containers

---

This section describes how to deploy **NVMesh 2.5.2 Management Servers** using containers. It is presumed that the administrator is familiar with such environments.

## Downloading the Docker image

- Obtain your username and password for the docker.excelero.com registry from [Excelero Technical Support](#). The following examples, use **John** for the username and **P@ssw0rd** for the password.
- Login to the registry: `docker login -u John docker.excelero.com.`
- Download the image, for instance: `docker pull docker.excelero.com/excelero/nvmesh-management:2.4.0-14.`
- Tag the image without the repository prefix: `docker tag docker.excelero.com/excelero/nvmesh-management:2.4.0-15 excelero/nvmesh-management:2.4.0-15.`
- The image can be downloaded to several machines or it can be downloaded to one node, saved as a .tar image, copied to all nodes and then loaded on each of them.
  - To save the image: `docker save excelero/nvmesh-management:2.4.0-15 -o ./nvmesh-management:2.4.0-15.tar.`
  - To load the image: `docker load -i ./nvmesh-management:2.4.0-15.tar.`

## Deploying NVMesh 2.5.2 Management Servers using Kubernetes

Before deploying, make sure you have a MongoDB instance or a replica set running and accessible from within the Kubernetes containers.

MongoDB can be installed on physical nodes or deployed in Kubernetes also.

### Deploying MongoDB in Kubernetes

Following are instructions for deploying MongoDB for in Kubernetes.

- Initial docker deployment: `kubectl -n nvmesh apply -f /opt/NVMesh/kubernetes/examples/mongo/mongo-rs.yaml`
- Provide persistent storage for the mongo pods using PVs. PVs with the label `role: mongo-data` are required.
  - To list the PVCs, use: `kubectl -n nvmesh get pvc`
  - **Note:** PVs of type local-storage will force the pod to run only on the specific host where the storage is available. If that node is down or unavailable, the pod will not run.
  - **Note:** It is recommended that persistentVolumes for different instances will not share a single point of failure. For example, if using NFS, use different NFS servers on different nodes instead of one NFS server on one node which will render the db unusable if this node fails. In addition, it may be prudent to use PersistentVolumes on storage with redundancy.
- Deploying with replica sets:
  - Change `spec.replicas` to the number of replicas required.
  - After the mongo pods have successfully started, run the ReplicaSet initialization command on one of the instances. For example, with a namespace of **management** to deploy on 3 hosts

named **mongo-1/2/3**:

- `kubectl exec -it -n management mongo-0 /bin/bash`
- `rs.initiate({ _id: 'rs0', version: 1, members: [ { _id: 0, host: 'mongo-0.mongo:27017' }, { _id: 1, host: 'mongo-1.mongo:27017' }, { _id: 2, host: 'mongo-2.mongo:27017' } ] });`

## Deploying a *Management* instance

- Create a new Kubernetes namespace: `kubectl create namespace nvmesh`.
- A PV is needed for database backups. Create them with the label `role: nvmesh-backups`. The **Management** will then make a PVC with a default size of 10GiB.
  - **Note:** PVs of type local-storage will force the pod to run only on the specific host where the storage is available. If that node is down or unavailable, the pod will not run.
  - **Note:** It is recommended that persistentVolumes for different instances will not share a single point of failure. For example, if using NFS, use different NFS servers on different nodes instead of one NFS server on one node which will render the db unusable if this node fails. In addition, it may be prudent to use PersistentVolumes on storage with redundancy.
- Edit and apply a ConfigMap appropriately for the environment.
  - Make a copy of the ConfigMap template, for instance: `cp /opt/NVMesh/kubernetes/examples/management/configmap.yaml ~/my-mgmt-config.yaml`.
  - Edit the following parameters in the `~/my-mgmt-config.yaml` file:
    - If you are using an Ingress controller with SSL Termination, the `UseSSL` parameter must be set to `false`.
    - For the Mongo connection string in the ConfigMap:
      - For a (bare-metal) host Mongo instance(s), use:
        - `config.mongoConnection.hosts = "<address>:<mongodb port>"`
        - `config.statisticsConnection.hosts = "<address>:<mongodb port>"`
        - Example for replica set connection: `mongoConnection.hosts = "nvme21:27017,nvme31:27017,nvme23:27017"`
      - For Kubernetes-based Mongo instance(s), use:
        - `mongoConnection.hosts = "<service-name>.<namespace>.svc.cluster.local:<mongodb port>"`
        - `statisticsConnection.hosts = "<service-name>.<namespace>.svc.cluster.local:<mongodb port>"`
- The field `statisticsIngressPort` determines the port on which nodes connect to the Ingress to send statistics to the pod. This is set by default to 443 and should match the incoming port of the ingress controller.
- Apply the configuration using `kubectl apply -n nvmesh -f ~/my-mgmt-config.yaml`.
- Deploy the **Management** services:
  - For a single management instance use: `kubectl apply -n nvmesh -f /opt/NVMesh/ku`

```
bernetes/examples/services/services-singel-mgmt.yaml.
```

- For a management replica use: `kubectl apply -n nvmesh -f /opt/NVMesh/kubernetes/examples/services/services-mgmt-replica.yaml.`
- Deploy the **Management StatefulSet**: `kubectl apply -n nvmesh -f /opt/NVMesh/kubernetes/examples/management/default.yaml.`

## Ingress configuration

Configuring the ingress is an important part of the configuration in Kubernetes. An improper configuration could lead to an unavailable or inaccessible service, network communication issues or faulty statuses may be presented.

Kuberentes supports 3rd party ingress controllers. Each controller requires some specific configuration that the Kubernetes ingress API does not support. This is usually done using annotations on the ingress object.

Following are instructions for the two most commonly used ingress controllers: K8s ingress-nginx and traefik ingress controller.

The following features should be available and configured in the 3rd party ingress controllers.

- Use or allow TLS.
- Support websocket connections.
- Configure maximum websocket connection timeouts.
- Support persistent sessions, i.e. sticky sessions. Implementations include cookie based sessions or clientIP based sessions. This is important mainly for GUI experience.

### Traefik Ingress Controller

- The following annotations should be added to the ingress object:

```
kubernetes.io/ingress.class: traefik
traefik.ingress.kubernetes.io/router.tls: 'true'
```

- The following annotations should be added to the nvmesh-management-gui service object:

```
traefik.ingress.kubernetes.io/service.sticky: 'true'
traefik.ingress.kubernetes.io/service.sticky.cookie.name: traefik-sticky
```

- To debug the Traefik ingress controller:
  - Edit the Traefik deployment, by adding `—log.level=DEBUG`
  - Check Traefik pod logs while running requests from a browser and using curl. Check for the existence of sticky session cookies when connecting to the GUI service.

## Kubernetes ingress-nginx Controller

**Note:** This is not the same as the [NGINX Ingress Controller for Kubernetes](#).

- The following annotations should be added to the ingress object:

```
nginx.ingress.kubernetes.io/affinity: cookie
nginx.ingress.kubernetes.io/affinity-mode: persistent
```

## Ingress Routing Rules

Best practices and examples for ingress routing rules can be found at `/opt/NVMesh/kubernetes/examples/ingress/`.

The ingress should forward each request to host `nvmesh-management-<n>` to its corresponding pod service to port `ws` and each `nvmesh-management-<n>-<m>` to its corresponding service to port `stats-<m>`. For more information, see the yaml examples below.

- For a replica of 3 managements: `kubectl apply -n nvmesh -f /opt/NVMesh/kubernetes/examples/ingress/replica-ingress.yaml`.
- For a single management instance using ingress-nginx: `kubectl apply -n nvmesh -f /opt/NVMesh/kubernetes/examples/ingress/single-mgmt-ingress.yaml`.
- For a single management instance using traefik: `kubectl apply -n nvmesh -f /opt/NVMesh/kubernetes/examples/ingress/single-mgmt-traefik.yaml`.

## Configuring the *Clients* and *Targets*

In `/etc/opt/NVMesh/nvmesh.conf` set the `MANAGEMENT_SERVER` appropriately for the **Management Servers** layout. The following is an example for 3 entry points, `MANAGEMENT_SERVER="nvmesh-management-0:443,nvmesh-management-1:443,nvmesh-management-2:443"`.

- For each replica instance, the node should resolve `nvmesh-management-` to the entryPoint IP configured for the ingress controller.
- For each replica instance and for each statistics port (the number of ports is defined in the config under `statisticsCores` and defaults to 5), the node should resolve `nvmesh-management--` to the entryPoint IP configured for the ingress controller, where `n` is the replica index and `m` is the statistics port index.
  - For a single **Management** instance with 5 statistics ports, `/etc/hosts` should contain:
    - `<ingress IP> nvmesh-management-0 nvmesh-management-0-0 nvmesh-management-0-1 nvmesh-management-0-2 nvmesh-management-0-3 nvmesh-management-0-4`
  - For a 3-way **Management** layout, again with the default 5 statistics ports, `/etc/hosts` should contain:
    - `<ingress IP> nvmesh-management-0 nvmesh-management-0-0 ... nvmesh-management-0-4 nvmesh-management-1 nvmesh-management-1-0 ... nvmes`

```
h-management-1-4 nvmesh-management-2 nvmesh-management-2-0 ... nvmesh-management-2-4
```

- To obtain the IP for the ingress, use `kubectl get service --all-namespaces | grep -e "ingress.*LoadBalancer` and look in the 5th column for an IP in the appropriate range.

## Connecting to the GUI

To access the **Management** GUI, use the ingress controller's external IP. If unsure what the externalIP of your ingress controller is, try the following command to list all externally exposed services in your cluster: `kubectl get service --all-namespaces | grep LoadBalancer`.

## Upgrading the *Management* in the future

Edit the object and rename the `spec.template.spec.containers.image` so that it points to the image version to upgrade to, use: `kubectl edit -n nvmesh statefulset nvmesh-management`.

## Multi-tenancy

Running multiple sets of **NVMesh 2.5.2 Management Servers** in the same Kubernetes cluster is supported. To do so, choose a unique namespace for each installation, i.e. replace `kubectl ... -n nvmesh ...` with `kubectctl ... -n <unique namespace> ...` for each system. Contact [Excelero Technical Support](#) for further clarifications.

# Deploying NVMesh 2.5.2 Management Servers in Docker

## Container Environment Variables

The following environment variables govern the Docker deployment.

### MONGO\_SERVERS

The addresses of the mongo servers in the format "addr1:port1,addr2:port2". This overrides `config.mongo.Connection.hosts` in `/etc/opt/NVMesh/management.js.conf`.

For example: `docker run .. -e MONGO_SERVERS="server1:27017,server2:27017"`.

### CONFIG

This is an optional configuration update object that will be applied to `/etc/opt/NVMesh/management.js.conf` in JSON Format.

For example: `docker run .. -e CONFIG='{ "mongoConnection" : { "options": { "replicaSetName": "rs0" } } }'`.

## FORCE\_IP

This is the IP other **Management Servers** will use to connect to this management instance. This IP on port 4001 should be accessible from other management instances. This overrides `config.forceIP` in `/etc/opt/NVMesh/management.js.conf`.

For example: `docker run .. -e FORCE_IP="10.0.0.50"`.

## Host-based Networking

```
CONFIG='{ "mongoConnection" : { "hosts": "localhost:27017" }, "statisticsMongoConnection": { "hosts": "localhost:27017" } }'
VERSION="2.0.2-76"
docker run -d --net nvmesh-br --name nvmesh-mgmt1 -e CONFIG="$CONFIG" -p 4000-4006:4000-4006 excelerero/nvmesh-management:$VERSION
```

Change to the specific **Management** version.

If `mongodb` is not on the same co-located, update `MONGO_SERVERS="localhost:27017"` with the appropriate mongo coordinates.

## Bridge-based Networking

- On the **Management** nodes:
  - Verify there is no MongoDB Server running on the host and that port 27017 is not in use.
  - Verify there is no **Management** instance already running on the host and that ports 4000 and 4001 are not in use.
- On the mongo nodes:
  - Create a local folder for the database. For instance, use `mkdir -p /data/nvmesh`. The directory can be any directory including on NFS mounted storage.
- To create the docker bridge network, use: `docker network create nvmesh-br`.
- Run a MongoDB 4.2 server container: `docker run -d --net nvmesh-br --name mongo -p 27017:27017 -v /data/nvmesh:/data/db mongo:4.2`.
- Run the following on each **Management** instance:

```
CONFIG='{ "mongoConnection" : { "hosts": "mongo:27017" }, "statisticsMongoConnection": { "hosts": "mongo:27017" } }'
VERSION="2.0.2-76"
docker run -d --net nvmesh-br --name nvmesh-mgmt1 -e CONFIG="$CONFIG" -p 4000-4006:4000-4006 excelerero/nvmesh-management:$VERSION
```

Set `VERSION` to the image version tag. i.e `excelero/nvmesh-management:1.3.2`. Use `docker images` to view the locally available images.

The default for the mongo server is localhost. If the mongodb and server containers are run on the same machine, this argument can be omitted.

## Example 1: Running Mongo Server and 3 Management Containers on the Same Host using docker network bridge

- On the host, create a local folder for mongo: `mkdir -p /data/nvmesh`.
- Create the docker bridge network: `docker network create nvmesh-br`.
- Run a MongoDB server container: `docker run -d --net nvmesh-br --name mongo -p 27017:27017 -v /data/nvmesh:/data/db mongo:4.2`.
- Run the **Management Servers**:

```
CONFIG='{ "mongoConnection" : { "hosts": "mongo:27017" }, "statisticsMongoConnection": { "hosts": "mongo:27017" } }'
VERSION="2.0.2-76"
docker run -d --net nvmesh-br --name nvmesh-mgmt1 -p 4000-4006:4000-4006 -e CONFIG="$CONFIG" excelero/nvmesh-management:$VERSION
docker run -d --net nvmesh-br --name nvmesh-mgmt2 -p 7000-7006:4000-4006 -e CONFIG="$CONFIG" excelero/nvmesh-management:$VERSION
docker run -d --net nvmesh-br --name nvmesh-mgmt3 -p 8000-8006:4000-4006 -e CONFIG="$CONFIG" excelero/nvmesh-management:$VERSION
```

## Example 2: Running Mongo Server and 3 Management Containers on Different Hosts

- On the mongo host, create a local folder: `mkdir -p /data/nvmesh`.
- Run the MongoDB server container: `docker run -d --name mongoddb-for-nrvmesh -p 27017:27017 -v /data/nvmesh:/data/db mongo:4.2 mongod --ipv6`
- On each **Management** node, run it:

```
CONFIG='{ "mongoConnection" : { "hosts": "<mongo-ip>:27017" }, "statisticsMongoConnection": { "hosts": "<mongo-ip>:27017" } }'
VERSION="2.0.2-76"
docker run -d --name nvmesh-mgmt --net host -e CONFIG="$CONFIG" excelero/nvmesh-management:$VERSION
```

## 4.5.3. Install the NVMesh Clients and Targets

### Installation

The **nvmesh-core** package contains both the **NVMesh 2.5.2 Client** and **Target** software. Install the **nvmesh-core** package on all the host(s) that require **NVMesh 2.5.2** logical block volumes (**Clients**) and the host(s) that have at least one NVMe drive (**Targets**).

To install the **NVMesh 2.5.2** software, run as root:

For Redhat/CentOS: `yum -y install nvmesh-core`

For Ubuntu: `apt install -y nvmesh-core`

\* In very small configurations, in which only 2 nodes contain NVMe drives, it may be necessary to install the software and enable the **Target** service on a third host without NVMe drives so that this host can be an arbiter for RAID 1/10 logical volumes.

### Configuration

For an initial installation, it is necessary to set the address or hostname of the **Management Servers** for the **Clients** and **Targets**. This performed in this configuration file:

```
/etc/opt/NVMesh/nvmesh.conf
```

All available configuration options for this file are detailed in the [Client and Target Options](#) section.

### Setting the Management

Within this configuration file, it is necessary to modify the `MANAGEMENT_SERVERS` variable. This can be accomplished by running a helper script, or by editing the configuration file directly.

#### Helper Script

Run the `nvmesh_configure_management_server` script provided by the **nvmesh-core** installation.

#### Direct File Manipulation

If you choose to edit the file manually, you'll be changing the `MANAGEMENT_SERVERS` variable:

```
MANAGEMENT_SERVERS="<hostname | IPADDRESS>:<port>, <hostname | IPADDRESS>:<port>, ..."
```

for example:

```
MANAGEMENT_SERVERS="172.10.100.201:4001"
```

The IP addresses (hostnames) should be that of the machine(s) on which **Management** was installed.

To limit **NVMesh 2.5.2** to use only some of the RDMA-capable NICs or to choose specific NICs in general, use the `nvmesh_configure_nics` CLI utility. Alternatively, see [Shared Configuration Options](#).

## 4.5.4. Enable RDDA

---

**!** Critical Update: In the final stages of 2.5 testing, an extremely rare issue that leads to data corruption was found with RDDA. Therefore, reinserting RDDA has been suspended until further notice. The remainder of this section is therefore obsolete for the time being.

### Overview

**!** It is imperative to validate the firmware version on **Targets** before turning on RDDA. Therefore, it is off by default. [Excelero Technical Support](#) will not be able to restore data if an erroneous firmware was used.

Support for RDDA was disabled in multiple versions of **NVMesh** as there was a potential issue with some of the NIC firmwares that could lead to data corruption. As the newer NIC firmwares have remedied these issues, it is now possible to enable RDDA.

To identify that your NIC firmwares do not have potential issues see the following information:

- Firmwares up to XXX.28.2006 work well.
- Firmwares XXX.29.\* could lead to data corruption.
- Firmwares XXX.30.\* could lead to data corruption.
- Firmwares XXX.31.\* and above work well.

Automatic firmware verification may be added in an upcoming release.

To enable RDDA, first install the **nvmesh-core** package, then perform the following procedure on all **Target** Mellanox ConnectX NICs.

### Enable RDDA in Mellanox ConnectX Adapters

Enabling RDDA on **Target** nodes provides the lowest latency and highest levels of performance, including **Target** side CPU offload.

**!** Enabling RDDA for ConnectX requires a one-time firmware option change. This change is necessary for **Target** nodes only.

To enable RDDA with Mellanox OFED:

- `sudo mst start`
- `sudo mst status` will return a list of adapters and other information. Identify the relevant adapters by product number.
- `sudo mlxconfig -d /dev/mst/<DEVICE>_pciconf0 -b /etc/opt/NVMesh/Excelero_mlxconfig.db set ONE_QP_PER_RECOVERY=1`
  - It is simplest to reboot the machine at this point. An alternative is to follow the guidance suggesting an online adapter reset.

To query RDDA support with Mellanox OFED:

- `sudo mlxconfig -d /dev/mst/<DEVICE>_pciconf0 -b /etc/opt/NVMesh/Excelero_mlxconfig.db query | grep "ONE_QP"`

To enable RDDA without Mellanox OFED, with the Inbox driver:

- `sudo mstconfig -d <DEVICE PCI ADDRESS> -b /etc/opt/NVMesh/Excelero_mlxconfig.db set ONE_QP_PER_RECOVERY=1`
- `mstconfig` is provide in the `mstflint` package. If this command use, `yum install mstconfig` to install it.

To query RDDA without Mellanox OFED, with the Inbox driver:

- `sudo mstconfig -d <DEVICE PCI ADDRESS> -b /etc/opt/NVMesh/Excelero_mlxconfig.db query | grep "ONE_QP"`

**\* NOTE:** Setting the above firmware parameter reduces the overall number of available queue pairs (QPs) to 64K for that adapter and all its physical ports and their functions (physical or virtual). This may not be desirable in very large environments.

**\* NOTE:** Performance with RDDA may also be affected by the number of virtual functions configured for the NIC. The actual recommendation here is NIC specific. Therefore, in general, it is recommended to keep the number of functions to a minimum. This is performed by setting the firmware `NUM_OF_VFS` variable to 1 or the desired minimum in the same way done above for `ONE_QP_PER_PER_RECOVERY`.

## Enable RDDA in the *Target's* configuration profile

Navigate to the {CONF\_PROFILES} tab located in the *Settings* menu. Locate the *Configuration Profile* for the *Target*. The *Target's* profile can be seen in the *Targets* tab. Edit the *Configuration Profile*, in *Advanced Options*, in the *Target* tab, enable this option.

If *NVMesh 2.5.2* was running, you must restart it on the node for the above setting to take effect.

## Enable RDDA in `nvmesh.conf`

Add the following to `/etc/opt/NVMesh/nvmesh.conf` to enable RDDA:

```
MLX5_RDDA_ENABLED="Yes"
```

If *NVMesh 2.5.2* was running, you must restart it for the above setting to take effect.

Setting this parameter to no will disable RDDA on the node. The setting in `nvmesh.conf` overrides the value from the *Configuration Profile*.

## 4.5.5. Configure TCP/IP Support

---

To configure an *NVMesh 2.5.2* node to utilize the TCP/IP network protocol on Ethernet NICs, perform the following modifications:

1. Edit `/etc/opt/NVMesh/nvmesh.conf`
  - a. If there is an entry for `TCP_ENABLED`, set it to `Yes` or add a line for this setting, as follows: `TCP_ENABLED="Yes"`.

The previous setting will enable simultaneous use of TCP/IP and RDMA. To disable RDMA entirely, perform the same operation as above with `TCP_ONLY`.

## 4.5.6. Start the NVMesh Client

---

### Starting the Client Service

After setting the *Management Servers* in the configuration file, start the *Client* service using `systemctl start nvmeshclient`

At this point, the *Client* should be active and report itself to the *Management Servers*.

## Automatic Startup

Upon successful installation of the **nvmesh-core** package, the **Client** will be set to automatically start at boot time.

## Verification

To verify that the service is running properly, use `systemctl status nvmeshclient`.

Additional information may be available from `journalctl -u nvmeshclient`.

## 4.5.7. Exclude Drives

---

### Overview

By default, all NVMe drives are automatically assigned to **NVMesh 2.5.2**, except drives that are already mounted such as boot drives. Additional drives can be excluded as well. The drives assigned to **NVMesh 2.5.2** are serviced by the **Target**, while the boot drives, mounted drives and the excluded drives will continue to be serviced by the standard kernel nvme module.

### Exclude Drives

**NVMesh 2.5.2** maintains the list of excluded NVMe drives in `/etc/opt/NVMesh/target_devices.conf`. This file can be modified using the `nvmesh_target` utility.

This file is only utilized by **Targets**. Its contents are essentially a list of Linux device paths and serial numbers.

To identify the drives serial numbers, list all NVMe drives:

```
sudo nvme list
```



The `nvme` command is provided by the `nvme-cli` package for most Linux distributions.

To exclude a specific drive from being assigned to **NVMesh 2.5.2** use:

```
nvmesh_target exclude nvme <nvme_serial>
```

To assign a specific drive to **NVMesh 2.5.2** that was previously excluded:

```
nvmesh_target include nvme <nvme_serial>
```

For example, drive SN `S3HCNX0K701049` should be excluded from NVMesh:

```
[root@nvme1013 15:25:26 ~]# nvme list
Node          SN          Model          Na
mespace Usage          Format          FW Rev
-----
/dev/nvme1000n1 S3HCNX0K600288 SAMSUNG MZWLL800HEHP-00003
1          800.17 GB / 800.17 GB 4 KiB + 8 B GPNA5B3Q
/dev/nvme1001n1 S3HCNX0K701270 SAMSUNG MZWLL800HEHP-00003
1          800.17 GB / 800.17 GB 4 KiB + 8 B GPNA5B3Q
/dev/nvme1002n1 S3HCNX0K600336 SAMSUNG MZWLL800HEHP-00003
1          800.17 GB / 800.17 GB 4 KiB + 8 B GPNA5B3Q
/dev/nvme1003n1 S3HCNX0K701048 SAMSUNG MZWLL800HEHP-00003
1          800.17 GB / 800.17 GB 4 KiB + 8 B GPNA5B3Q
/dev/nvme1004n1 S3HCNX0K701049 SAMSUNG MZWLL800HEHP-00003
1          800.17 GB / 800.17 GB 4 KiB + 8 B GPNA5B3Q
[root@nvme1013 15:26:56 ~]# nvmesh_target exclude nvme S3HCNX0K701049
nvme serial: S3HCNX0K701049 was added to excluded list
[root@nvme1013 15:27:02 ~]# cat /etc/opt/NVMesh/target_devices.conf
nvme,S3HCNX0K701049,0x144d,SAMSUNG MZWLL800HEHP-00003,1
```

\* If the excluded NVMe drive was already bound to **NVMesh 2.5.2**, the host must be rebooted for the drive to be excluded.

## 4.5.8. Start the NVMesh Target

### Start the Target Service

After setting the **Management Servers** in the configuration file, start the **Target** service using `systemctl start nvmeshtarget`

\* If the service status is failed, you may need to reset `systemd` using the `systemctl reset-failed` command.

At this point, the **Target** should be active and report itself, available NVMe drives and NICs, to the **Management Servers**. The reported NVMe drives will be visible at the **Management** level. The drives assigned to **NVMesh 2.5.2** will become available to the shared storage pool after being formatted.

## Automatic Startup

The **Target** is not set to automatically start at boot time. To set the service to start automatically use `systemctl enable nvmeshtarget`

## Verification

To verify that the service is running properly, use `systemctl status nvmeshtarget`. Additional information may be available from `journalctl -u nvmeshtarget`.

## 4.5.9. Add non-NVMe Drives

**NVMesh 2.5.2** excels when using NVMe drives as the storage media. Non-vanilla NVMe drives, such as SATA SSD drives or virtual NVMe-over-Fabrics connected block devices, can also be used as storage media.

To use such devices, perform the following operations:

- Enable non-NVMe support on the **TOMAs** for all or relevant **Targets**, using `/opt/NVMesh/common-repo/tools/toma_rpc config generic_block_device_support 1`.
  - Setting 0 instead will disable it.
- Restart the **Target** for this change to take effect.
- Format the drives, see [Format Drives](#) for more details.

 The add/remove block device options provided in the `nvmesh_target` utility are no longer relevant for non-NVMe drives. Do not use these commands.

## 4.5.10. (Optional) Assign Targets to Zones

For a description of **Zones** and their utility, see [Zones](#).

To enable zoning, stop all **Management Servers** and change the line `config.enableZones = false;` to `config.enableZones = true;` in `/etc/opt/NVMesh/management.js.conf`. Then, restart all **Management Servers**. The best indicator that the functionality is working is a new column **Zone** in the **Targets** section. There should also be a new button named **Approve**.

To assign **Targets** to a **Zone**, use the table's multi-select functionality and click the **Approve** button. In the pop-up dialog, fill in the zone-id, a positive integer, for the **Targets**, click **Approve** and then confirm in the **Targets** section.

Once volumes have been allocated on the *Drives* on a **Target**, it will not be reassignable to a different zone, as this may lead to data loss.

## 4.5.11. Format Drives

---

Drives must be formatted for use with NVMesh 2.5.2.

**!** This operation is destructive and will cause loss of access to any data previously written to the drive.

To format a single drive using the GUI:

1. Click **Targets**.
2. Click the target that contains this drive.
3. Click **Format** in the desired drive.
4. Enter the login password.
5. Click **OK** to confirm the format operation.

Alternatively, and to format multiple drives at once, using the GUI:

1. Click **Drives**.
2. Select the drives requiring a format, or mark the top-most checkbox to select all drives.
3. Click **Format**.
4. Enter the login password.
5. Click **OK** to confirm the format operation.

# 5. Getting Started

A simple workflow is presented for configuring an **NVMesh 2.5.2** system for the first time. This workflow introduces the **Management Servers** GUI, an innovative web-based interface for managing the system.

The workflow consists of the following steps:

- Introduction to the **Management GUI**
- Managing drives
- Managing volume groups
- Managing volumes
- Attaching volumes
- Resizing volumes

## 5.1. Drives Management

### Overview

The *Drives* screen lists all the drives in the system and provides the ability to format them.

### Drives Table

Target	SN	Vendor	Model	Capacity	Block Size	Metadata Size	Status	Excluded	Evicted	Health
scale-1.excelero.com	35CNSZV175QY0	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-1.excelero.com	35CNSZV175QY1	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-1.excelero.com	35CNSZV175QY2	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-1.excelero.com	35CNSZV175QY3	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-1.excelero.com	35CNSZV175QY4	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-1.excelero.com	35CNSZV175QY5	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-1.excelero.com	35CNSZV175QY6	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-1.excelero.com	35CNSZV175QY7	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-1.excelero.com	35CNSZV175QY8	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-1.excelero.com	35CNSZV175QY9	Intel	INTEL SSDPE2ME400G4	360GB/395.91GB (91%)	4KB	8B	Ok			✓
scale-10.excelero.com	RPTMNV0COCU6.0	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-10.excelero.com	RPTMNV0COCU6.1	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-11.excelero.com	D9QC9AXW876W.0	Intel	INTEL SSDPE2ME400G4	360GB/395.91GB (91%)	4KB	8B	Ok			✓
scale-11.excelero.com	D9QC9AXW876W.1	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-12.excelero.com	P22U247HXMS4.0	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-12.excelero.com	P22U247HXMS4.1	Intel	INTEL SSDPE2ME400G4	360GB/395.91GB (91%)	4KB	8B	Ok			✓
scale-13.excelero.com	5KM47FS2632Q.0	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-13.excelero.com	5KM47FS2632Q.1	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-14.excelero.com	G1M3W63CRWR6.0	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓
scale-14.excelero.com	G1M3W63CRWR6.1	Intel	INTEL SSDPE2ME400G4	08/395.91GB (0%)	4KB	8B	Ok			✓

The *Drives* table provides the following details (in the Status column) on drives detected by the system. The Health column presents additional explanatory information.

Status	Meaning
<b>Uninitialized</b>	The drive is not initialized for <b>NVMesh 2.5.2</b> usage and cannot be used by, until it is formatted. Drives that are in use by other software, such as a boot drive, will be automatically excluded. Formatting will not be possible for such drives.
<b>Ingesting</b>	The drive is in the process of being moved from the stock NVMe driver to the <b>NVMesh 2.5.2</b> driver. This operation should be rapid and will rarely be seen.
<b>Frozen</b>	The drive is in the initial formatting stage. This operation should be rapid and will rarely be seen.
<b>Formatting</b>	The drive is being actively formatted and making progress.
<b>Format_Error</b>	A format error occurred during the format operation.
<b>Initializing</b>	The drive has completed formatting and blocks are now being initialized, i.e. zeroed. The drive can be used for logical volume definition, but may not yet be ready for a volume to come online if the blocks used by the volume have not been initialized yet. Initializing progress is presented as a percentage as it is reported from the <b>Target</b> .
<b>Excluded</b>	The drive had been administratively excluded from <b>NVMesh 2.5.2</b>
<b>Ok</b>	The drive was formatted by <b>NVMesh 2.5.2</b> and is available for use.
<b>Error</b>	An error occurred during the initialization of the drive.
<b>Missing</b>	The drive is currently missing from the cluster.

## 5.2. Volume Creation and Attachment

After completing installation of the **Management**, **Client** and **Target** software on selected hosts, the next step is volume definition. For the example, the following configuration will be used:

Hostname	# of Drives	Management	Client	Target
<b>nvmeshmgr1</b>	0	Yes	No	No
<b>appsvr1</b>	0	No	Yes	No
<b>appsvr2</b>	0	No	Yes	No
<b>appsvr3</b>	0	No	Yes	No
<b>targetsvr1</b>	2	No	No	Yes
<b>targetsvr2</b>	2	No	No	Yes

targetsrv3	2	No	No	Yes
------------	---	----	----	-----

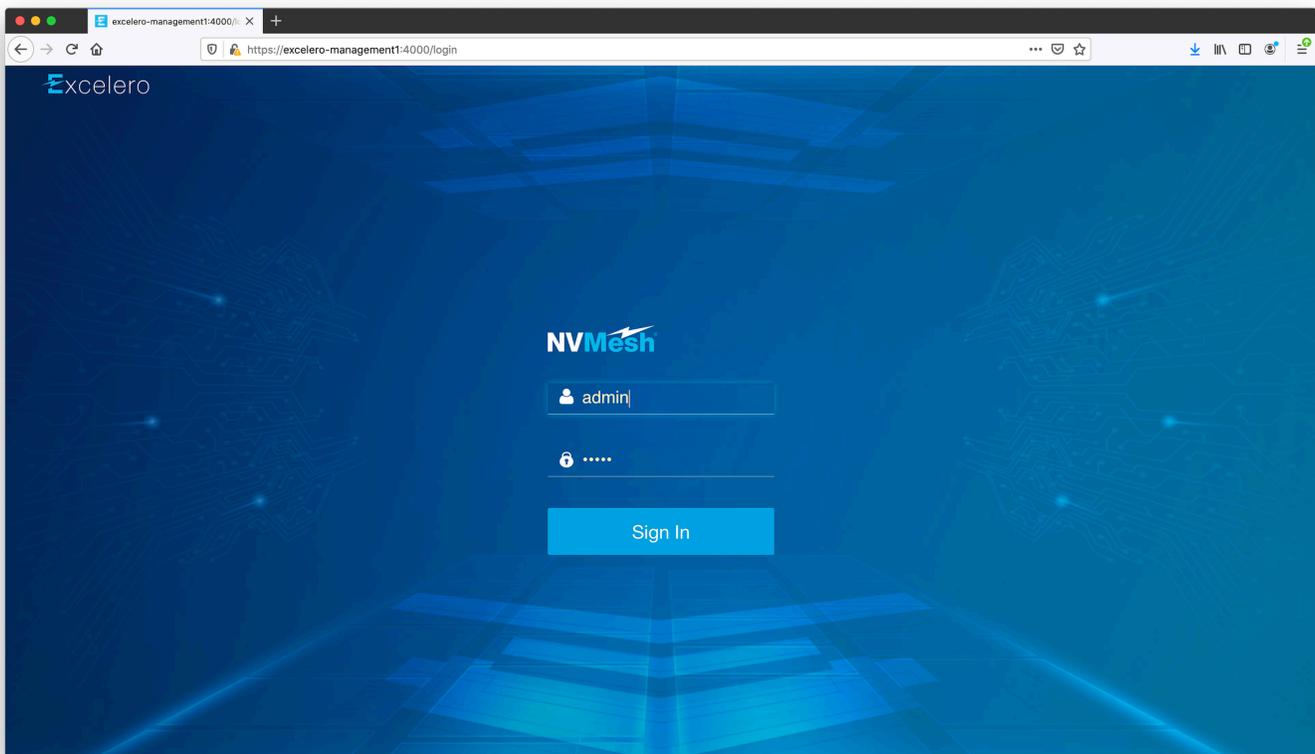
This represents a disaggregated **NVMesh 2.5.2** implementation with 3 hosts acting as **Clients**, 3 as **Targets** and nvmeshmgr1 as the sole **Management**.

Note that although targetsrv1, targetsrv2 and targetsrv2 are marked above as **Targets**, the **nvmeshclient** service is required for the **nvmeshtarget** service and will be run on these nodes, which could also behave as clients.

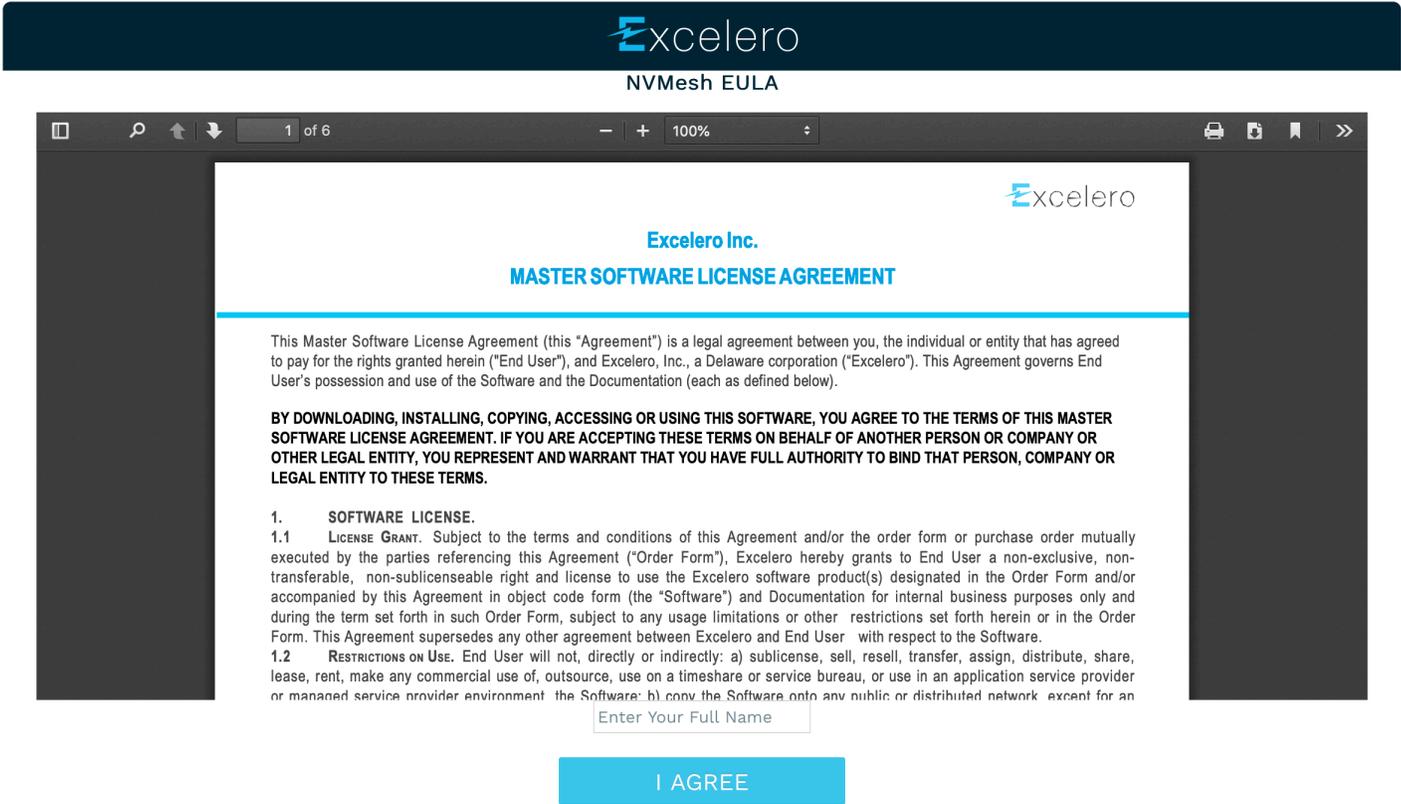
## 5.2.1. Login to the Management GUI

By default, the management server comes with a single default account/password, admin/admin

Login to the **Management** from a browser substituting its IP address or hostname. For example, `https://nvmeshmgr1:4000`

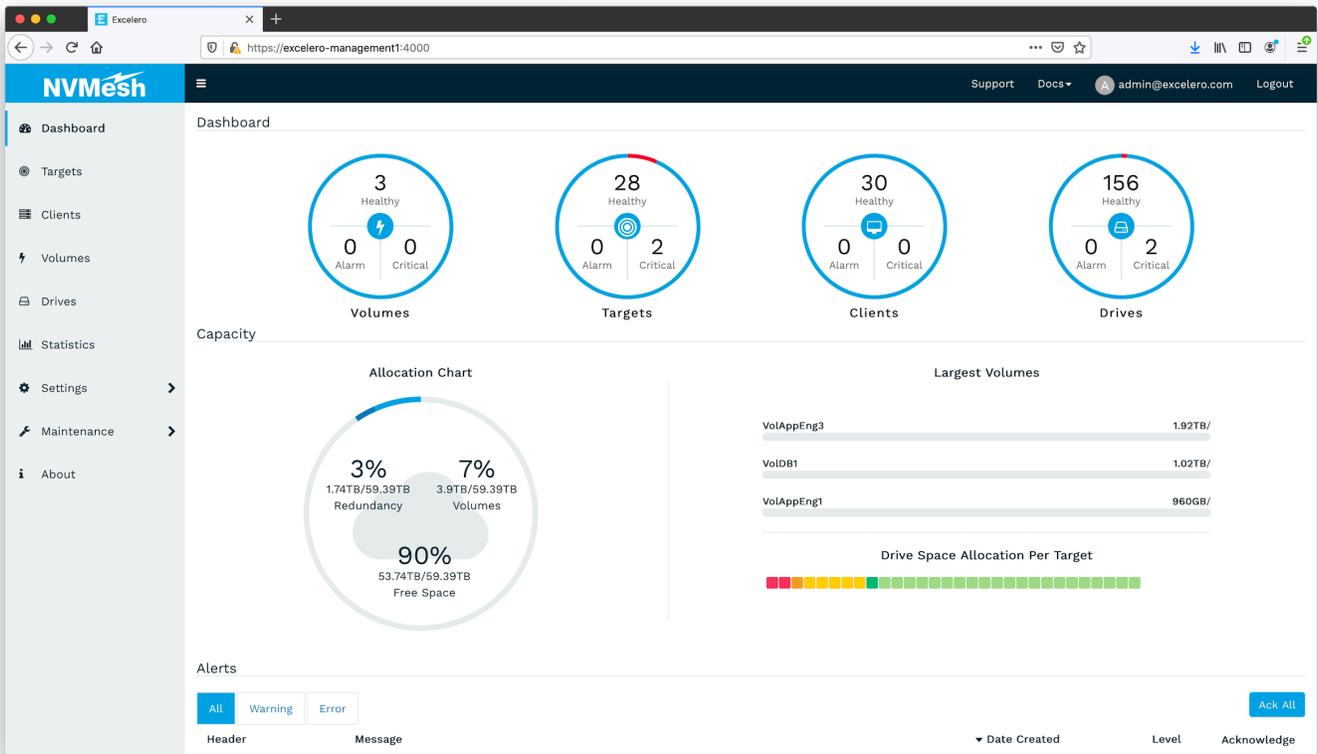


Upon initial login you will be prompted to accept the Excelero End User License Agreement (EULA) and to change the default password.

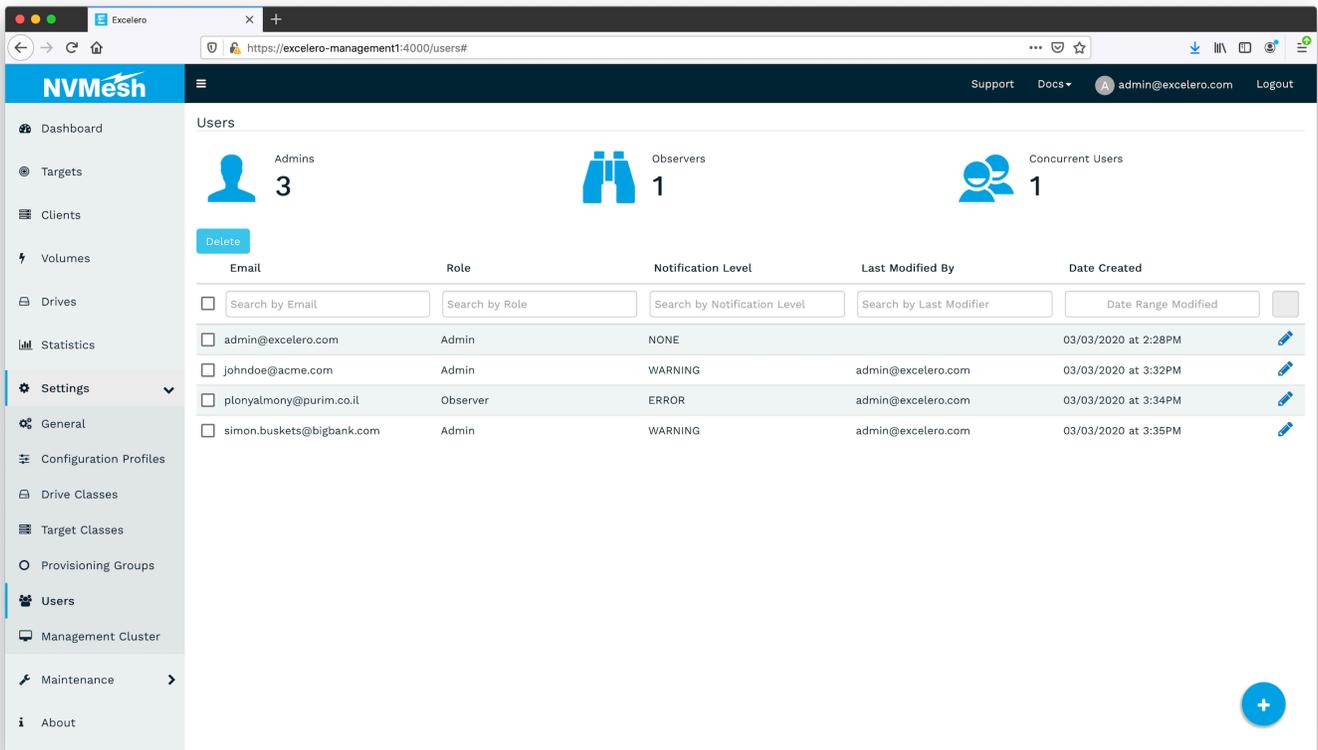


Read the agreement. Once you have scrolled down to the bottom, the button will become clickable. Enter your full name to accept the agreement.

After login (and potentially changing the admin password), the *Dashboard* view appears:



At this point, it is suggested to create a new administrative user. To do so, click *Settings* and then *Users* from the left side menu to see this screen:



Click the + sign button in the lower right corner to add a new user per the example below. The password set should be considered temporary as the GUI will ask to change it upon initial login by the new user.

## User ✕

**Email**

**Password**

**Re-type password**

**Role**

**Email Notifications Level**

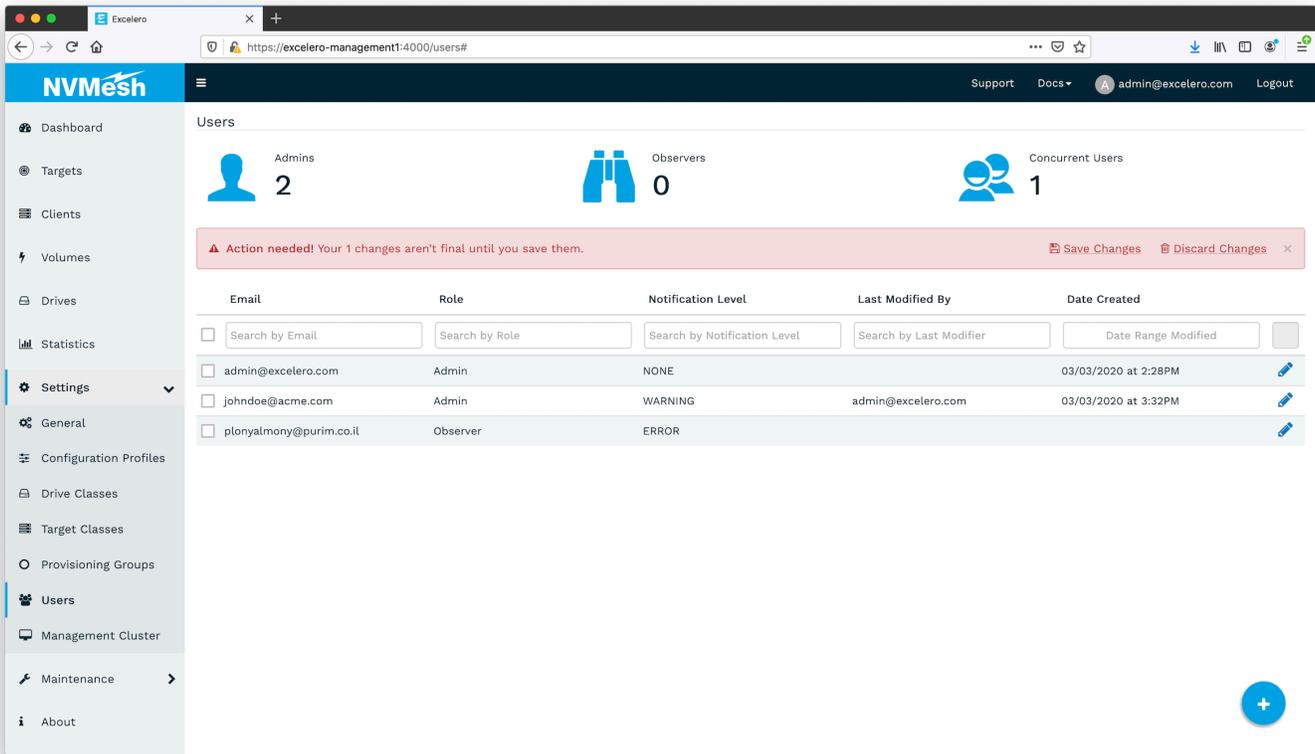
**Add** **Cancel**

The *Dialog Box* options are as follows:

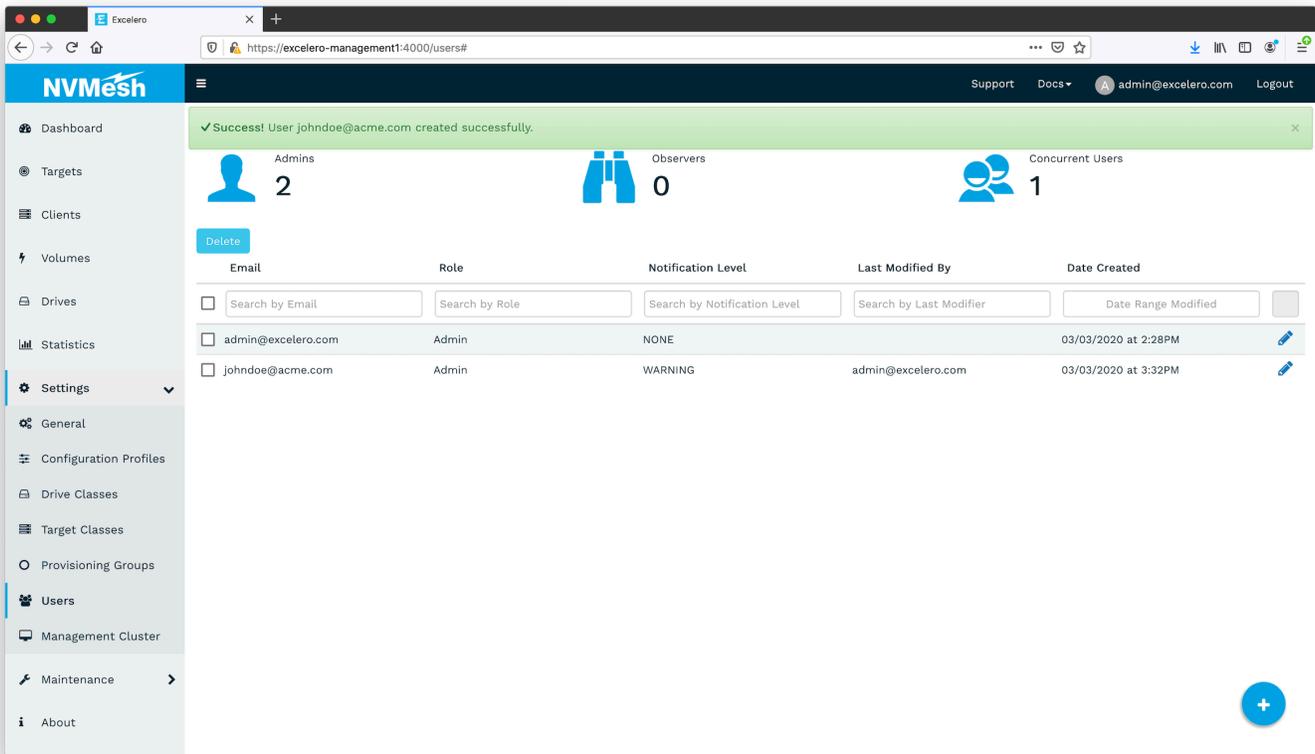
- **Email**
  - User Input: Enter a valid Email address as an account login
- **Password**
  - User Input: Enter a (temporary) password
- **Role**
  - Pulldown List:
    - *Admin*: Admin accounts can make all changes to the system including creating other

- accounts.
  - *Observer*: Observers can only monitor the system and cannot make any changes.
- **Email Notifications Level**
  - Pulldown List
    - *NONE*: No messages will be sent to the account.
    - *WARNING*: System Warnings and Errors will be sent to the account.
    - *ERROR*: System Errors will be sent to the account.

After clicking the *Add* button, confirm or commit the changes by clicking the *Save* button or the *Save Changes* link.



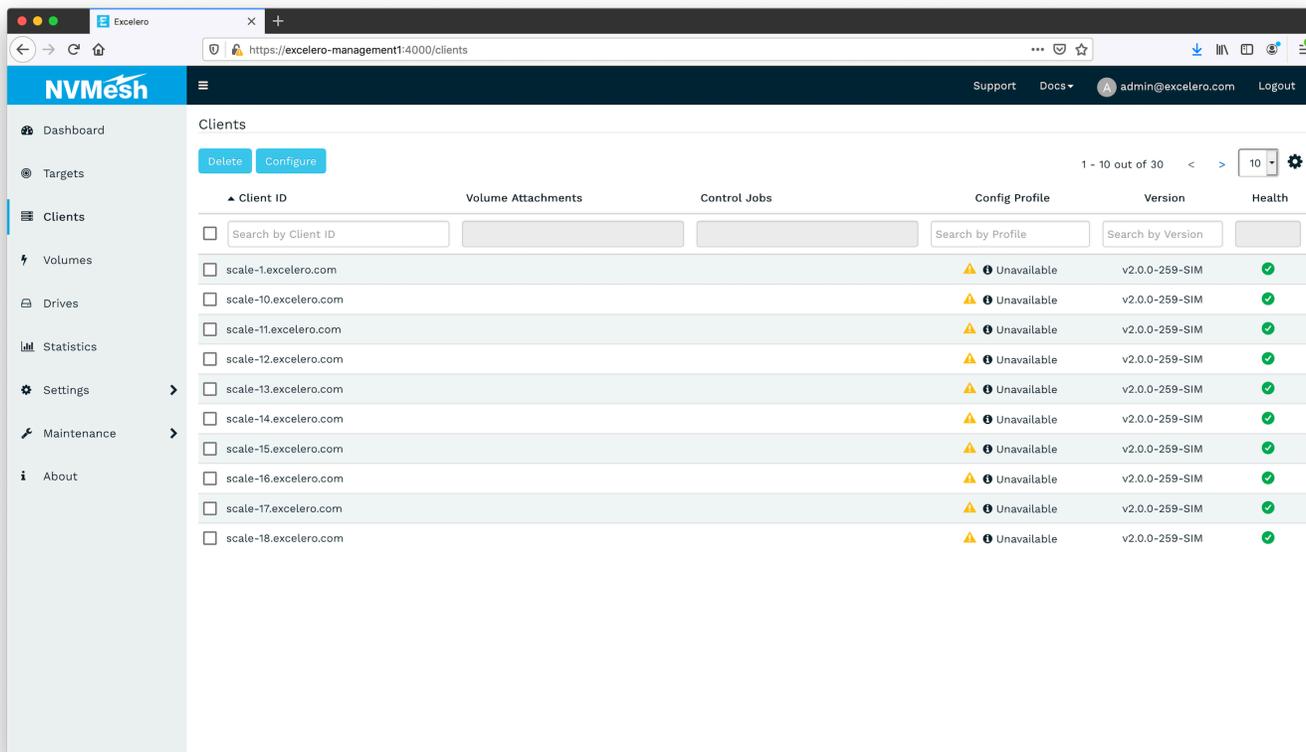
Changes in **NVMesh 2.5.2** require final confirmation by clicking *Save* or changes will be lost!



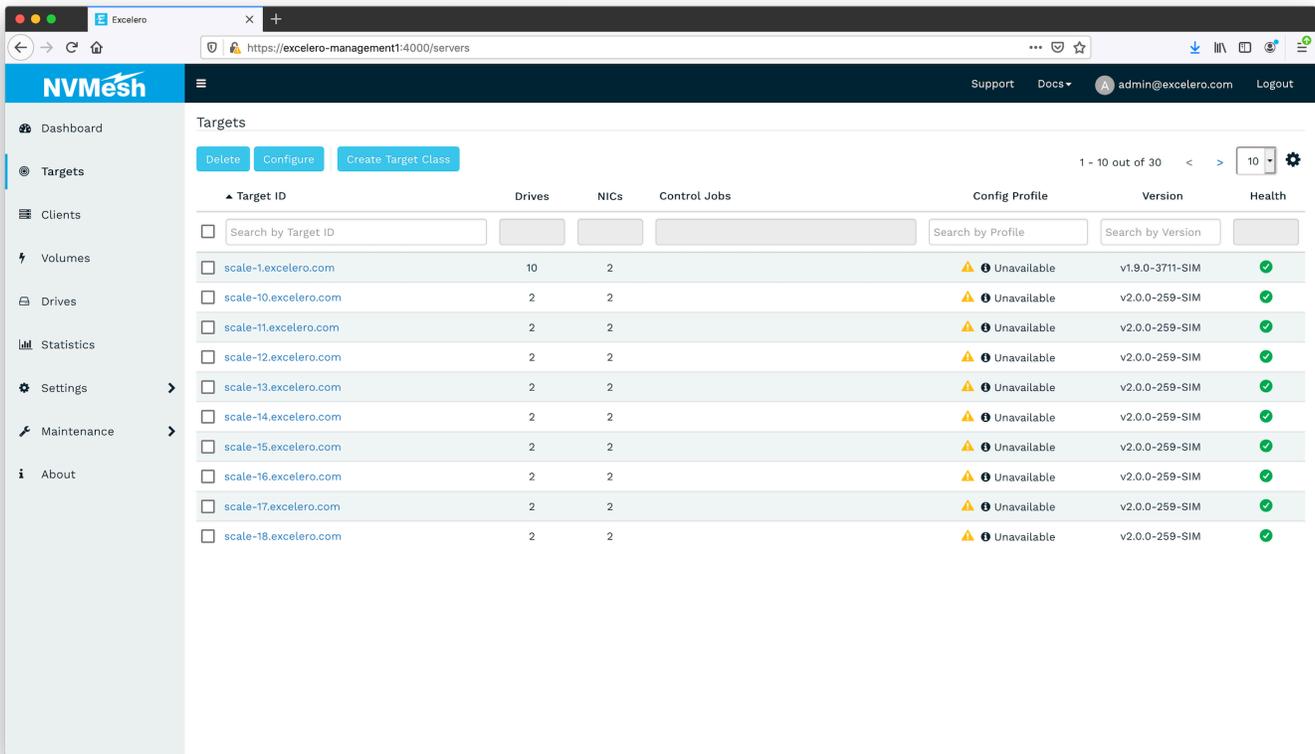
After successfully adding the new user, logout using the button in the top right corner and then log back in as the newly created user.

## 5.2.2. Verify Client and Target Registration

In the example, there are 3 *Clients* and 3 *Targets*. Choose the *Clients* menu item on the left to see a screen similar to:



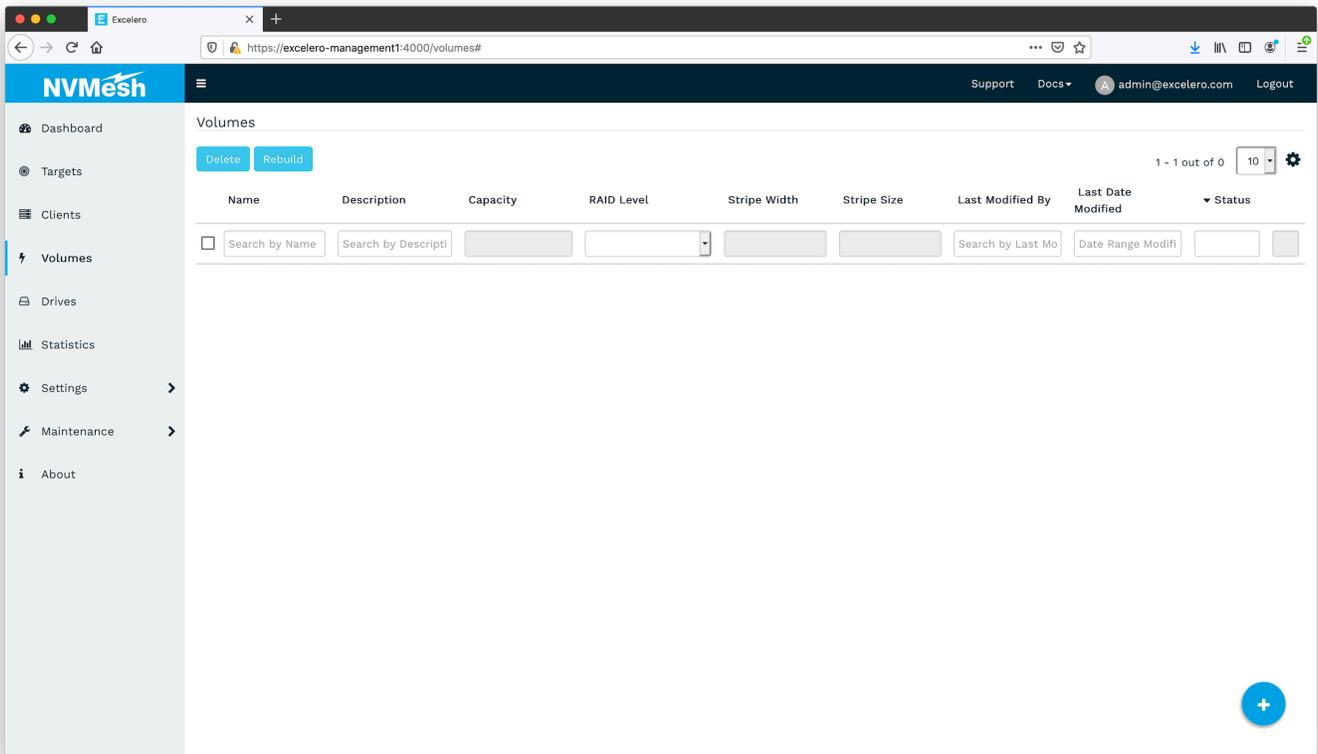
For all nodes on which the **nvmesh-core** package has been installed and the **nvmeshclient** service started, there should be a row in the *Clients* table. Since there are no *Volumes* yet, there is not much to do now other than verifying that the active **Clients** appear. Similarly, clicking the *Targets* menu item should show a screen similar to:



Clicking on a **Target** name will present that node’s NICs and NVMe devices. Feel free to explore, but the main objective at this stage is to simply verify the appearance of the **Targets** and that their status is OK.

## 5.2.3. Create a Volume

After verifying that the **Management** is functioning and that the **Clients** and **Targets** are reporting to the **Management**, a **Volume** can be created. Once created, it can be attached to **Clients** for use. Click the **Volumes** menu item on the left side of the screen for the following screen. Then, click the **+** sign to add a new **Volume**.



After clicking the button, a new dialog box opens prompting for the attributes of the new volume. In this example, a 400GB RAID-1 (mirrored) volume will be created.

Volume
✕

**Name**

**Description**

**Volume Capacity**

Maximum Available: 59.39TB

GB

Maximum, as much as possible

Volume Provisioning Group Custom

**Volume Provisioning Group**

99.97% Free

Add

Cancel

The *Volume* dialog box options are as follows:

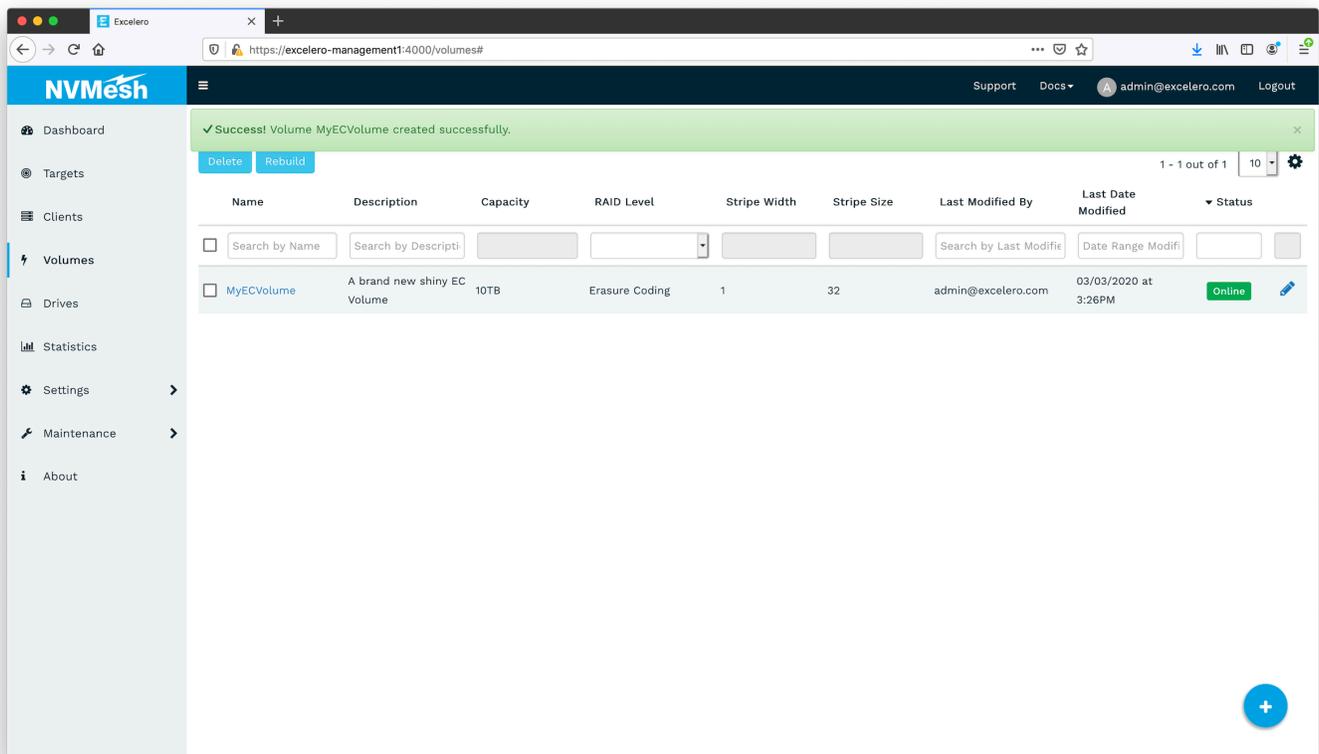
- **Name**
  - This should be a short name without special characters. This will be used as the block device name on the client in `/dev/nvmesh/...`
- **Description**
  - This is an optional human readable description of the volume.
- **Volume Capacity**
  - Specify the size of the volume or select *Maximum, as much as possible* to generate the largest possible volume that meets the provisioning criteria.
  - **Unit Type**
    - Choose the volume size units.
- **Volume Provisioning Group** or **Custom** Tab
  - Volume Provisioning Groups allow specifying drive selection criteria in an automated fashion. This topic is covered in the [Functionality Reference section](#).
  - Custom allows specifying all volume definition settings manually.

After entering or selecting options in the *Name*, *Description* and *Volume Capacity* fields, select the `DEFAULT_RAID_1_VPG` *Volume Provisioning Group*. A complete description of *Volume Provisioning Groups* can be found in the reference section [Volume Provisioning Groups](#). **NVMesh 2.5.2** will select areas of available space on NVMe devices that meet the necessary criteria for the selected *Volume Type*. In this example, it will utilize 400GB of space from one drive on one **Target**, and 400GB of space from a drive on a different **Target**. Mirrored halves must reside on different **Targets**.

The usage bar at the bottom of the dialog box displays the proposed capacity usage based on the currently specified options. To complete the volume creation, click the *Add* button.

 Changes must be saved by clicking the *Save* button or the *Save Changes* link. Otherwise, the new *Volume* will not be created.

Upon successful creation, a screen similar to this will appear:



After the successful creation of a *Volume*, it can be attached to *Clients* for consumption.

## 5.2.4. Attach a Volume to a Client

*Volumes* must be attached to *Clients* so they can be used. When attaching a *Volume* to a *Client*, it is made available as a block device named `/dev/nvmesh/<vol_name>` and operates like a regular Linux block device. To attach the *Volume* created in the previous section to a *Client*, log in to the *Client*, `appsvr1` in this example, to attach `testvol` to and at the shell prompt, with root permission:

```
nvmesh_attach_volumes testvol
```

For more information on the command line attach tool, see [Attaching Volumes](#).

Verify that the Linux block device `/dev/nvmesh/<vol_name>` has been created, as follows:

```
sh-4.2$ ls -l /dev/nvmesh/testvol
brw-rw---- 1 root disk 259, 2 Jun 25 04:52 /dev/nvmesh/testvol
```

To verify minimal block device functionality and attributes, use `fdisk`, as follows:

```
sh-4.2# fdisk -l /dev/nvmesh/testvol
```

```
Disk /dev/nvmesh/testvol: 400.0 GB, 399999369216 bytes, 97656096 sectors
Units = sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 131072 bytes
```

See the [Checking Volume Status](#) section for methods to get detailed status information on the **NVMesh 2.5.2** block device.

The block device is ready to be used just like any local NVMe device. You can create a file system on it or use it in a raw fashion. On subsequent service or system restarts, the device will automatically be attached if it was attached at service stop time and the configuration profile for this **Client** is defined to perform auto-attach. To prevent this behavior, explicitly detach the device via the management server or command line tool, as follows:

```
nvmesh_detach_volumes testvol
```

## 6. General Settings

Click **Settings** and then click **General** to reach the set of general settings governing various aspects of *NVMesh 2.5.2* behavior.

### General Settings

Sub-section	Setting	Values
Cluster Statistics	Collect Statistics	Governs whether the <b>Management Servers</b> will collect performance data from the endpoints.
	Limit Collection	A regular expression or list of nodes can be defined here. This will limit statistics collection from these nodes only.
Security	Multi-tenancy	Governs whether multi-tenancy security restrictions are applied in the system. If they are, volumes must be associated with <i>Volume Security Groups</i> and security keys must be distributed to <b>Clients</b> to enable successful attachment of volumes. See <a href="#">Securing Volume Attachments</a> for more information.
Cluster ID	Cluster ID	An ID given by the administrator to describe the cluster.
Unit Types	Unit Types	When decimal is chosen, the convention for volume and drive size will be decimal based, i.e. using kilobytes, megabytes, gigabytes, terabytes, etc. For binary, the convention used will refer to kibibytes, mebibytes, gibibytes, tebibytes, etc.
Default Domain	Default Domain	The default email domain to use for users that do not supply a domain name explicitly.
Customer Name	Customer Name	The customer name to which <a href="#">Excelero Technical Support</a> should associate notifications from this system.
Send Data to Excelero	Send Logs	Specifies the severity of logs sent to Excelero.
	Send Statistics	Determines whether to periodically send statistics to Excelero.
Advanced	Automatic Log Out Threshold	The timeout of the GUI and API access (in milliseconds). After the timeout expires the GUI and API will automatically logout all logged in users.
	“Keep Alive” Grace Period	The grace period, since the last message from every component. When the grace period is over, the management server will consider that component as timed out.
	Maximum JSON Size	The size of the largest JSON message supported by <b>Management Servers</b> .

	Reserved Blocks	The percentage of reserved blocks at the start of a managed NVMe device.
	Compatibility Mode	Use the <b>NVMesh 2.5.2</b> version of dynamic libraries instead of the operating system versions to avoid compatibility issues.
	Enable Legacy Formatting	Determines whether to allow legacy formatting on metadata supported drives via the RESTful API.
	Enable Volumes Access Via NVMe – System Default	The default value used for new volumes with regard to enabling access via the NVMe protocol.
Zones	Enable	Determines whether the <i>Zones</i> functionality is globally enabled in the system. See <a href="#">Zones</a> .
	Randomness	The amount of randomness to insert into <i>Zone</i> selection for Volume allocation. Should be between 0 and 100 percent. Default is 20.
	Selection Weights	
	Number of Segments in Zone	The weight to attribute to the number of segments already in a <i>Zone</i> . A higher value of segments decreases the chance the <i>Zone</i> will be chosen. Default is 150.
	Number of Targets in Zone	The weight to attribute to the number of <b>Targets</b> in a <i>Zone</i> . A larger number of targets increases the chance the <i>Zone</i> will be chosen. Default is 120.
	Average Time in Zone Allocation Queue	The time weight to attribute to time spent in allocation. A larger amount of time spent decreases the chance the <i>Zone</i> will be chosen. Default is 50.
Logging	Logging Level	The logging level of <b>Management Servers</b> .
	Days Before Log Entry Expires	The number of days the management logs are kept before rotation.
Statistics	Send Statistics Interval	The frequency at which the “phone home” statistics should be sent to <a href="#">Excelero Technical Support</a> .
	Cache Update Interval	The frequency at which management information caches are updated across the system. It is recommended not to configure without consulting <a href="#">Excelero Technical Support</a> .
	Statistics Report Interval	The frequency of statistics updates from the endpoints to <b>Management Servers</b> .

	Full Client Report Interval	Normally, <b>Clients</b> only send updates of changes to <b>Management Servers</b> . This setting sets the frequency of a full report.
User Settings	Unit Types	Sets the choice of unit types for the currently logged in user, which may be different than the value set under the general Unit Types setting, see above.

# 7. Client and Target Configuration

The guided installation example enables the deployment of a basic **NVMesh 2.5.2** environment. The following section serves as a reference for additional and advanced options that may be necessary or desired due to redundancy requirements, media selection or security requirements.

## 7.1. Configuration Profiles

*Configuration Profiles* introduced in **NVMesh 2.5.2** are used to make configuration changes for multiple **Clients** and **Targets** centrally.

Each node is associated with one *Configuration Profile*.

Upon installation, there are 3 built-in *Configuration Profiles* that cannot be deleted, as follows:

1. **NVMesh Default**, a read-only *Configuration Profile* that is equivalent to the default configuration after a fresh installation of **NVMesh 2.5.2**.
2. **NVMesh Debug**, a *Configuration Profile* that can be used for troubleshooting the system in which debug log messages are turned on.
3. **Cluster Default**, this is the *Configuration Profile* for any new node added to the cluster. Initially, its contents are the same as **NVMesh Default**. However, as this is editable, it may differ from **NVMesh Default** over time.

All configuration elements can also be configured through configuration files on the nodes themselves. Those options are described in the following sections.



Configuration elements defined locally on nodes using other means described in the subsequent sections will override settings from *Configuration Profiles*.

Nodes can be associated with *Configuration Profiles* by editing the *Configuration Profile* or by selecting nodes in either the **Clients** or **Targets** screen and pressing the **Configure** button, which will generate a new *Configuration Profile* unless all chosen nodes are exactly the nodes associated with a specific *Configuration Profile*.

## Configuration Elements

### Standard Options

Domain	Name	Description	Default Value
Cluster-	<b>Management</b>	A comma separated list of the <b>Management Servers</b> , in the form	Empty

Wide	<b>Servers</b>	<Management server hostname or IP address : port>, <hostname/IP:port>,... <ul style="list-style-type: none"> <li>Typically, the port number is 4001.</li> </ul>	
	<b>Management Protocol</b>	The web protocol for internal communication to <b>Management Servers</b> , either http or https	https
Client	<b>Auto Attach Volumes</b>	With auto-attach, <b>Volumes</b> are re-attached upon <b>Client</b> restart	Enabled

## Advanced Options

Domain	Name	Description	Default Value
Node	<b>Configured NICs</b>	A list of the NICs that <b>Clients</b> and <b>Targets</b> can use for IO, in the form < Interface Name: Transport Type; Interface Name : Transport Type >. An empty value places no limits.	Empty
	<b>IPv4 Only</b>	Do not use IPv6 for RoCEv2 (RDMA) or TCP networking for IO	Disabled
	<b>Maximum SM Query Burst</b>	Used to set the maximum burst size of queries to the IB Session Manager. This parameter is not relevant for RoCE. A smaller number here will decrease the load on the SM, but may increase the initial bring-up time.	Empty
	<b>TCP Enabled</b>	Enable TCP as a possible transport type for <b>Clients</b> and <b>Targets</b>	Disabled
Target	<b>Enable RDDA</b>	Used to enable RDDA for ConnectX adapters	Disabled
Logs	<b>Dump ftrace on kernel PANIC</b>	Used to dump the fast log (ftrace) buffers on kernel panic. Useful for debugging.	Disabled
	<b>MCS Logging Level</b>		INFO
	<b>MCS Logging Verbose Types</b>	Change only with instructions from Excelero Customer Support	Empty
	<b>Management Agent Logging Level</b>		INFO

## 7.2. Client and Target Options

Most **Client** and **Target** options reside in the same file:

`/etc/opt/NVMesh/nvmesh.conf`

While the configuration file is shared by the **Client** and **Target**, some options within the file are applicable only to one or the other.

 Options specified in this configuration file only take affect when the associated services are started, or restarted.

## 7.2.1. Client and Target Shared Options

The following options are used by **Clients** and **Targets**:

### CONFIGURED\_NICS

#### Description:

Limit the NICs to be used for **NVMesh 2.5.2**. By default, **Clients** and **Targets** will attempt to make use of any RDMA capable NIC. If populated, only the NICs in the list will be utilized.

#### Default Value:

CONFIGURED\_NICS="" (Empty)

#### Possible Values:

A semi-colon separated list of NIC identifiers in the following format:

CONFIGURED\_NICS="<INTERFACE>;<INTERFACE>;<INTERFACE>; . . ."

### IPV4\_ONLY

#### Description:

Only support IPv4 for RoCEv2/TCP

 TCP/IP NVMesh functionality does not currently operate on IPv6 based networks and hence IPV4\_ONLY should be set to "Yes" for TCP.

#### Default Value:

IPV4\_ONLY="No"

#### Possible Values:

"Yes" or "No"

## IPV6\_ONLY

### Description:

Only support IPv6 for RoCEv2/TCP



Setting both IPV4\_ONLY and IPV6\_ONLY to "Yes" will render an error in service startup.

### Default Value:

```
IPV6_ONLY="No"
```

### Possible Values:

"Yes" or "No"

## MANAGEMENT\_PROTOCOL

### Description:

Specifies the communication protocol between the *Clients* and *Targets* and the *Management Servers*.

### Default Value:

```
MANAGEMENT_PROTOCOL="https"
```

### Possible Values:

"http" or "https"

## MANAGEMENT\_SERVERS

### Description:

Addresses or hostnames and port of the optionally redundant *Management Servers*. The port should match that of

### Default Value:

```
MANAGEMENT_SERVERS="nvmesh-management:4001"
```

### Possible Values:

A comma separated list of *Management Servers* and ports in the following format:

```
MANAGEMENT_SERVERS="<hostname | IPADDRESS>:<port>,<hostname | IPADDRESS>:<port>,..."
```

**Example:**

```
MANAGEMENT_SERVERS="nvmeshmgr1:4001,nvmeshmgr2:4001,nvmeshmgr3:4001"
```

**MAX\_SM\_QUERY\_BURST****Description:**

For InfiniBand only, the maximum numbers of queries per second to send to the subnet manager.

**Default Value:**

```
MAX_SM_QUERY_BURST="32"
```

**Possible Values:**

An integer value

**TCP\_ENABLED****Description:**

Enable usage of TCP over Ethernet NICs.

**Default Value:**

```
TCP_ENABLED="No"
```

**Possible Values:**

"Yes" or "No"

**TCP\_ONLY****Description:**

Enable usage **only** of TCP over Ethernet NICs, effectively disabling RDMA.

**Default Value:**

```
TCP_ONLY="No"
```

**Possible Values:**

"Yes" or "No"

## MCS\_MANAGEMENT\_TIMEOUT

### Description:

Allows detecting a hanging/dead TCP connection between **Clients** and **Targets** and the **Management Servers**, before the TCP timeout (controlled by `tcp_retries2`) is reached. The timeout defaults to 10 mins or more on some machines.

### Default Value:

```
MCS_MANAGEMENT_TIMEOUT="30"
```

### Possible Values:

Any whole number of seconds between 30-600.

## COLLECT\_STATS

### Description:

Allows shutting off and on the statistics samplers from the **Management Servers**. ManagementAgent will turn on statistics collections when it is enabled on the **Management**. This option allows overriding statistics collection for specific **Clients** and **Targets**.

### Default Value:

```
COLLECT_STATS="Yes"
```

### Possible Values:

"Yes" or "No"

## 7.2.2. Client Specific Options

---

The following options are relevant only for **Clients**:

## AUTO\_ATTACH\_VOLUMES

### Description:

Determines whether or not previously attached volumes are re-attached automatically after system or **Client** startup.

### Default Value:

```
AUTO_ATTACH_VOLUMES="yes"
```

**Possible Values:**

“yes” or “no”

**Example:**

```
AUTO_ATTACH_VOLUMES="no"
```

**ISCSI\_INITIATOR\_IP****Description:**

Sets the the IP of the ISCSI initiator remote node

**Default Value:**

```
ISCSI_INITIATOR_IP=""
```

**Possible Values:**

Valid IP addresses

**Example:**

```
ISCSI_INITIATOR_IP="192.168.1.1"
```

**ISCSI\_TARGET\_IP****Description:**

Sets the the IP of the ISCSI target node using this format: <IP>:<PORT>.

**Default Value:**

```
ISCSI_TARGET_IP=""
```

**Possible Values:**

Valid IP addresses and a port separated by a colon.

**Example:**

```
ISCSI_TARGET_IP="192.168.1.1:1234"
```

## NVMF\_IP

### Description:

Defines the set of IP addresses listened on for **NVMe-oF** exposure of volumes.

### Default Value:

```
NVMF_IP=""
```

### Possible Values:

This should be filled out as a semi-colon separated list of valid IP addresses. Leaving this value blank will make the system unable to provide **NVMe-oF** access to volumes and render an error in the log.

### Example:

```
NVMF_IP="192.168.1.1;192.168.1.2"
```

## 7.2.3. Target Specific Options

---

The following options are relevant only for **Targets**:

### MAX\_CLIENT\_RSRC

#### Description:

Determines the number of RDDA queue pairs (QPs) to be assigned, per client, per *target* drive. Can be used to limit IO to a *target node*'s drives as a rudimentary QoS setting. Setting this value to 0 (zero) effectively turns off RDDA.

#### Default Value:

Not set/present. This means the default behavior when a client connects to a *target* drive is to take all available QPs.

#### Possible Values:

An integer value from 0 (zero) to the maximum number of drive QPs

#### Example:

```
MAX_CLIENT_RSRC="128"
```

## MLX5\_RDDA\_ENABLED

### Description:

For Mellanox ConnectX-4 and later model adapters. Determines whether RDDA will be used on the *target* to offload CPU.

! ConnectX-4 model adapters may require a firmware update to enable RDDA. ConnectX-5 and later model adapters ship with the pre-requisite firmware level by default.

\* It is important to note that in addition to turning on RDDA, for best performance (lowest latency), it is also required to enable a specific setting in the adapter firmware. For more information refer to [NIC Requirements](#).

### Default Value:

```
MLX5_RDDA_ENABLED="No"
```

### Possible Values:

"Yes" or "No"

### Example:

```
MLX5_RDDA_ENABLED="Yes"
```

## NVME\_IRQ\_AFFINITY\_DOMAIN

### Description:

Defines how the interrupts for **NVMesh 2.5.2** managed NVMe drives are distributed to CPU cores. The optimal choice is dependent on the machine's NUMA architecture and the number of NVMe queues used.

### Default Value:

pernuma

### Possible Values:

- *none* : Controlled by the irqbalance service
- *pernuma* : Round-robin among cores of the NUMA node of the PCI root complex to which the NVMe drive is connected.
- *persocket* : Round-robin among cores of the physical CPU (socket) of the PCI root complex to which

the NVMe drive is connected.

- *fullspread* : Round-robin among all cores in the system.

**Example:**

```
NVME_IRQ_AFFINITY_DOMAIN ="fullspread"
```

## 7.3. Module Parameters

For the most part, options are configurable via the `/etc/opt/NVMesh/nvmesh.conf` configuration file as described in the [Client and Target Options](#) section.

There are some parameters that are configurable as module parameters. These are described in the following table. These parameters can be managed using standard module parameter management tools, such as configuration files in `/etc/modprobe.d` for Enterprise Linux distributions.

 In general, these should be considered advanced options and end-users should not make changes to these options unless instructed to do so by [Excelero Technical Support](#).

### Common Parameters for *Clients* and *Targets*

Module	Parameter Name	Description
nvmeib_common_public	config	Binary tracer engine configuration. Consult <a href="#">Excelero Technical Support</a> before changing.
nvmeib_common_public	hide_warnings_stack	Hide warnings from dmesg, the kernel log, while keeping them in the binary traces. Enabled by default.
nvmeib_common_public	ib_odp_info	Defines whether On-Demand-Paging is enabled. 1 – Enabled, 0 – Disabled, -1 – Auto.
nvmeib_common_public	ipoller_poll_duration_jif	ipoller poll duration till reschedule, 0=default (uint). This is used for shared completion queue polling.
nvmeib_common_public	num_warnings	The number of warnings the module has triggered. Contact support if this is not 0.
nvmeib_common_public	tracer_dbg_level	Internal.
nvmeib_common_public	tracer_dbg_mask	Internal.
nvmeib_common_public	tracer_debug_level	Control path tracing debug level [0..4]. Default is 1.
nvmeib_common_public	tracer_wq_debug_level	Internal.

nvmeib_common_public	wq_max_processing_time	Maximum wq processing time before doing rescheduling self in jiffies. Default is 3000.
nvmeib_common	cq_vec_flags	CQ completion-vector selection flags. A value of b0 : Reserve vec 0 (for userspace). A value of b1 : Index based (set by cq creator). A value of b2 : Same vector for SCQ/RCQ. Consult <a href="#">Excelero Technical Support</a> before changing.
nvmeib_common	cq_vec_flags_tcp	CQ completion-vector selection flags. A value of b0 : Reserve vec 0 (for userspace). A value of b1 : Index based (set by cq creator). A value of b2 : Same vector for SCQ/RCQ. Consult <a href="#">Excelero Technical Support</a> before changing.
nvmeib_common	cq_vec_snd_rcv_delta	CQ completion-vector – delta between SCQ and RCQ Vector (ignored for same vector SCQ/RCQ)
nvmeib_common	cq_vec_snd_rcv_delta_tcp	CQ completion-vector – delta between SCQ and RCQ Vector (ignored for same vector SCQ/RCQ)
nvmeib_common	debug_level	Defines whether to output all dynamic logs. 2 or greater – Yes, 1 or less – No.
nvmeib_common	ib_cross_subnet	Enable cross subnet, IB transport (bool). Used to disable or enable attempts to communicate across IB subnets. Default is false.
nvmeib_common	ipv6_mode	IPv6 Mode: 0 – No IPv6, 1 – IPv6 enabled, but prefer IPv4 addresses (default), 2 – IPv6 enabled and preferred, 3 – IPv6 Only
nvmeib_common	json_iostats_fixed_size	Defines whether to pad I/O statistics in JSON representation to a fixed size.
nvmeib_common	numa_alloc_granularity	Defines the number of pages on the same NUMA node before traversing to the next one. Default is 64.
nvmeib_common	numa_alloc_policy	Defines the memory allocation policy per NUMA for large allocations. 0 – kernel-defined, usually per-chance where the allocating thread was run. This is the default value. 1 – Round robin on all NUMA nodes. 2 – Round robin on nodes within the same CPU socket as the PCI device associated with the allocation.

nvmeib_common	num_warnings	The number of warnings the module has triggered. Contact support if this is not 0.
nvmeib_common	pcpu_cq2srq_size_margin	When employing a per CPU shared completion and receive queue, this determines how much bigger the SCQ is than the SRQ. Margin = CQ-size – SRQ-size. Default is 1024.
nvmeib_common	pcpu_cq_comp_vecs_per_dev	Y – use first N comp-vectors of device where N='max num of pcpu-cqs per device' ; N – use all comp-vectors spread globally between all devices (usually not recommended) (bool). Default is true.
nvmeib_common	pcpu_cq_flush_del_qps	percpu cqs flush QPs pending for deletion (bool). Only for internal use.
nvmeib_common	pcpu_cq_intr_budget	percpu cqs interrupt-mode's budget (uint). This is the maximum number of completions to handle in a single interrupt. Default is 4.
nvmeib_common	pcpu_cq_max_cqs_per_dev	Maximum number of percpu cqs per device. If set to 0, use system default (uint). Default is 0.
nvmeib_common	pcpu_cq_poll_budget	percpu cqs polling-mode's budget (uint). This is the maximum number of times to poll for a completion. Default is 256.
nvmeib_common	pcpu_cq_size	The length or size of the shared completion queue when employing a per CPU shared completion and receive queue. Default is 4096.
nvmeib_common	qp_retry_cnt	QP retry count. Default is 7.
nvmeib_common	qp_timeout	QP timeout ( $4.096 \times 2^N$ ) us. Default (N) is 14.
nvmeib_common	tracer_debug_level	Control path tracing debug level [0..4]. Default is 1.

## Client Parameters

Module	Parameter Name	Description
nvmeiba	verbose_debug	Defines logging level.
nvmeibc	cfg_id	Client configuration profile ID
nvmeibc	cfg_name	Client configuration profile name.
nvmeibc	cfg_version	Client configuration profile version.

nvmeibc	debug_level	Defines whether to output all dynamic logs. 2 or greater – Yes, 1 or less – No. Deprecated.
nvmeibc	default_dir_lsblk	Defines the directory within /dev in which <b>NVMesh 2.5.2</b> block devices will be generated. Can be left empty to use /dev. Default value: <code>nvmesh</code> .
nvmeibc	disk_locks_fast_reuse	Reuse disk-lock opr before calling ulp cb (bool). This is a new optimization. Default is false.
nvmeibc	elect_destage_delay_seconds	Experimental for future functionality.
nvmeibc	elect_max_destages_per_dev	Experimental for future functionality.
nvmeibc	elect_torn_write_protection	Experimental for future functionality.
nvmeibc	force_reconf_reboot	Force all block device configuration changes to be done via device reboot, i.e. restarting the block device. Default is false.
nvmeibc	goodpath_debug_level	Data path tracing debug level. Default is 2.
nvmeibc	goodpath_locks_debug_level	Data path locks tracing debug level. Default is 3.
nvmeibc	goodpath_transport_debug_level	Data path transport tracing debug level. Default is 1.
nvmeibc	guids	Used for port filtering functionality.
nvmeibc	io_alloc_dma_key	Allow use of a DMA key for IO Channel to improve performance. Default is true.
nvmeibc	ioch_ka_only_no_rdda	Use no-RDDA channels for IO keep-alives. Note that all current such channel won't be affected until reattach/relogin. Default is true.
nvmeibc	jam_max_used_entries	Jam max used journal-entries, if 0 use system default (uint). Default is 0.
nvmeibc	jam_non_free_entry_timeout	Jam timeout for jentry in non-free state [sec] [sec] (ulong). Default is 300.
nvmeibc	jam_pending_enb	Jam pending mode control switch.

		Default is true (Y).
nvmeibc	jam_pending_req_timeout_jif	Jam timeout for pending allocation request [jiffies], if 0 use system default (ulong). Default is 10 milliseconds.
nvmeibc	lock_ch_get_by_cpu_index	Choose the lock channel to the disk by the CPU core used for non-TCP transports. Default is N.
nvmeibc	lock_ch_get_by_cpu_index_tcp	Choose the lock channel to the disk by the CPU core used for TCP transports. Default is Y.
nvmeibc	lock_retry_delay_multiplier	Lock retry timeout.
nvmeibc	management_report_frequency	Report frequency for management, in seconds. Default is 30.
nvmeibc	map_each_sg_entry	IB DMA map each sg-entry separately. Default is false.
nvmeibc	map_sg_mode	Mapping data SG list modes: 0 (default): Combined, use global key when able to collapse all sg-entries to one, otherwise map-mr (map WR and rdma-write WR) 1: Use only map-mr (IB_WR_REG_MR, IB_WR_FAST_REG_MR) 2: Use only global dma key, may use multiple rdma-write WRs from clnt/srv in wr/rd, respectively. Target must have enough WRs to write back on read-op
nvmeibc	map_sg_result_trace	Trace result of map data SG list: 0: Disabled, 1: One-shot (edge), 2: Continuous (level)
nvmeibc	max_ioch_rm_works	Max concurrent IO communication channel removal operations. Default is the number of cores in the server.
nvmeibc	max_ios_per_cpu	Maximum number of concurrent IO operations handled per core. Can be used to prevent IO flooding. In other words, the upper limit on the number of outstanding IOs to issue via the block driver per CPU core. Some file systems and applications queue or

		<p>perform read-ahead very aggressively, likely to overcome problems with legacy storage solutions. With <b>NVMesh 2.5.2</b>, large numbers of outstanding read requests may lead to network congestion especially when target bandwidth exceeds client bandwidth. Throttling the number of outstanding requests using this parameter can reduce this congestion and improve overall quality of service. Limiting this value often ends up improving performance for the <b>Client</b> and others on the network. If in doubt, start with a value of 8.</p> <p>This setting can be applied dynamically to the kernel module without restarting services.</p> <p>Default is 64.</p>
nvmeibc	max_lock_channels	The maximum number of lock channels for non-TCP transports. Default is 5.
nvmeibc	max_lock_channels_tcp	The maximum number of lock channels for TCP transports. Default is 16.
nvmeibc	max_nic_srqs	Maximum number of shared receive queues to define per NIC. Default is 16, which should provide sufficient performance for most use cases
nvmeibc	max_rcomp_intr	Max number of recv completions to handle in an interrupt before entering poll mode. Default is 64
nvmeibc	max_trim_size_mirrored	Maximum size of a single NVMesh internal TRIM operation for mirrored volumes. Default is 128, which translates to 16 MB
nvmeibc	max_trim_size_non_mirrored	Maximum size of a single NVMesh internal TRIM operation for non-mirrored volumes. Default is 256, which translates to 32 MB
nvmeibc	mini_elevator	Enable mini-elevator which combines writes to erasure coded volumes to fill

		stripes. Default is false.
nvmeibc	module_fast_log_enabled	Controls fast logging for data path. 1 – enable, 0 – disabled. Default is 0.
nvmeibc	no_part_scan	Disable partition scan on nvmesh block devices. Default is false.
nvmeibc	nr_defer_recv_comps	Defer processing of No-RDDA received completions for non-TCP transports. Default is true.
nvmeibc	nr_defer_recv_comps_tcp	Defer processing of No-RDDA received completions for TCP transports. Default is false.
nvmeibc	nr_get_by_cpu_index	Choose No-RDDA channel by CPU core used for non-TCP transports. Default is false.
nvmeibc	nr_get_by_cpu_index_tcp	Choose No-RDDA channel by CPU core used for TCP transports. Default is true.
nvmeibc	nr_get_least_used	Get least used No-RDDA channel, improves performance in certain scenarios (bool). Default is false.
nvmeibc	nr_max_channels_per_disk	Maximum No-RDDA channels per disk. A minimum of 4 channels for drives or disks with metadata is forced. Default is 64.
nvmeibc	nr_max_channels_per_path	Maximum No-RDDA channels per disk per networking path. Default is 4.
nvmeibc	nr_max_channels_per_path_tcp	Maximum No-RDDA channels per disk per networking path for TCP. Default is 16.
nvmeibc	nr_max_used_reqs_per_channel	Maximum number of requests issued simultaneously on a channel. Default is 64, can be increased up to 96.
nvmeibc	nr_pcpu_channels	Connect per-cpu No-RDDA channels (up to 128) on top of the nr_max_channels_per_disk any-cpu channels. The total number of No-RDDA channels is limited by nr_max_channels_per_path of both

		client and target. Default is false.
nvmeibc	nr_rotate_in_pending	Rotate No-RDDA channel list when reusing channel for pending commands, improves performance in certain scenarios (bool). Default is false.
nvmeibc	nr_shared_cq	No-RDDA channels use a shared completion queue (SCQ) for RDMA. Default is true.
nvmeibc	nr_shared_cq_tcp	No-RDDA channels use a shared completion queue (SCQ) for TCP. Default is false.
nvmeibc	nr_skip_rdma_write	<b>This is an unsafe debug mode:</b> No-RDDA IOs skip the RDMA write for write operations. This will always work on Legacy volumes and on EC when CRC check is off and bs=slice_length_of_volumes but not store the right data. Used for performance testing. Default is false.
nvmeibc	nr_wd_long_timeout	No-RDDA channel's watchdog timeout in jiffies. Default is 0, which means 10 seconds.
nvmeibc	nr_wd_rescue_timeout	No-RDDA channel's watchdog rescue timeout in jiffies. A value under 1 second disables this functionality. Default is 1500, i.e. 1.5 seconds.
nvmeibc	num_warnings	The number of warnings the module has triggered. Contact support if this is not 0.
nvmeibc	nvmeibc_copy_bio_buffers	Copy bio buffers in writes. Set when using page cache (bool). Default is true.
nvmeibc	nvmeibc_debug_ram_binfo	Enforce detection of topological data corruptions in RAM. Default is true.
nvmeibc	nvmeibc_default_debug_di	Upon attach, enable debug di mode. Default is false. This should not be changed without <a href="#">Excelero Technical Support</a> guidance.
nvmeibc	nvmeibc_elect_active_md_cache_max_lru_size_per_dev	Experimental for future functionality.

nvmeibc	nvmeibc_jentry_num_blocks	Length of erasure coding journal, in blocks. Default is 1.
nvmeibc	nvmeibc_jmd_wr_version	Version of JMD to use, internal. This should not be changed without <a href="#">Excelero Technical Support</a> guidance.
nvmeibc	nvmeibc_sync_full_lockset_probability_factor	Defines the probability of sync'ing 128K blocks instead of the current IO size.
nvmeibc	nvmeibc_sync_max_operations_per_dev	Maximum number of outstanding sync operations per volume.
nvmeibc	pages_max_alloc	Maximum order of page allocations allowed. Default is 31.
nvmeibc	panic_on_core_dbgdi	On core-dbgdi detection panic both client and target (bool). An internal debugging facility. Default is false.
nvmeibc	per_cpu_lock_transfer_num	Number of lock transfer candidates per CPU. Default is 8.
nvmeibc	ports	Used for port filtering functionality.
nvmeibc	profiling_enabled	Enable statistics gathering, should be 0 if clocksource != tsc. Default is true.
nvmeibc	qa_ec_stress_debug	For QA only! Over stress EC datapath. Default is false
nvmeibc	rdda_pois_bb	Fills the remote buffer with a special value to avoid memory corruption errors. This is a 2nd level of protection. Default is true.
nvmeibc	recovery_iterator_cooldown	Recovery iterator timeout to wait after completing full recovery cycle in jiffies. Default is 0.
nvmeibc	resub_aware_throttle_sleep_ms	Resubmit thread awake throttle sleep time in milliseconds. Default is 10. Should be in the order of scheduler process switching.
nvmeibc	resub_aware_throttle_threshold_ms	Resubmit thread awake throttle threshold in milliseconds. 0 – throttle disabled. Default is 1000.
nvmeibc	self_recovery_detach_time_sec	Time-out for idle hidden volume until

		<p>self detached in seconds.</p> <p>Hidden volumes are ones to which the client connects to perform recovery operations. Default is 5.</p>
nvmeibc	skip_disk_iocmds_flags	<p><b>This is an unsafe debug mode:</b> Skip disk access (remote and local) – 0 (default) : Disabled</p> <p>1: Skip Read operations</p> <p>2: Skip Write operations</p> <p>3: Skip Read &amp; Write operations</p> <p>4: Skip Jour-Writes or any combination using an operator</p> <p>5: Skip All IO Operations – even ones not mentioned here.</p> <p>Used for performance tuning and debugging.</p>
nvmeibc	skip_lock_cmds_flags	<p><b>This is an unsafe debug mode:</b> Skip lock cmds for non-EC volumes (remote and local) -</p> <p>0: Disabled</p> <p>b0: cmp_exchange</p> <p>b1: active lock table</p> <p>b2: read lock</p> <p>b3: write binfo</p> <p>Used for performance tuning and debugging.</p>
nvmeibc	sm_th	<p>Maximum number of concurrent Infiniband subnet manager requests.</p> <p>Used for throttling subnet manager access. Default is 32.</p>
nvmeibc	spread_resub_work	<p>Spread lock-transfers to other cores. 0=No, 1=SysWq. Default is 0. The other option often improves throughput for serial write workloads.</p>
nvmeibc	tcp_mode	<p>Activate TCP transport mode, filter out RoCE devices. Default is false.</p>
nvmeibc	topology_debug_level	<p>Topology path tracing debug level. Default is 5.</p>
nvmeibc	tracer_debug_level	<p>Control path tracing debug level. Default is 4.</p>

nvmeibc	unprotected_write_period_seconds	Timeout in seconds for an unprotected volume until it is become read only. Default is infinite, functionality is deprecated.
nvmeibc	use_local_bypass	Access local drives directly and not via a NIC. Default is true.
nvmeibc	use_norrrda_for_io	Allow using No-RDDA channels for IO. Default is 1.
nvmeibc	use_only_norrrda_for_io	Allow using only No-RDDA channels for IO. Default is 0.
nvmeibc	use_pcpu_cq	Use a per CPU shared completion queue (SCQ) and shared receive queue (SRQ). Default is false.
nvmeibc	use_rdda	Allow using RDDA for the client. Default is 1.
nvmeibc	warn_if_lock_took_more_than_n_msec	If lock acquisition takes more than this value in milliseconds, issue a warning to the log. Default is 20,000.

## Target Parameters

Module	Parameter Name	Description
nvmeibs	cap_transfer_size	Cap all disks' max-transfer-size to 128KB. Default is true.
nvmeibs	debug_level	Defines whether to output all dynamic logs. 2 or greater – Yes, 1 or less – No. Deprecated.
nvmeibs	defer_recv_comps	Defer no-RDDA receive completions. For internal use. Default is false.
nvmeibs	distr_intr_program	Path to the interrupt distribution program for NVMe device interrupts. Default is /usr/bin/nvmesh_set_irq_affinity.
nvmeibs	dummy_id	Serial ID to be used for drives on drive-less targets.
nvmeibs	fake_serial	Fake serial number for NVME drive (should be machine specific). Default is empty.
nvmeibs	goodpath_debug_level	Data path tracing debug level. Default is 2.
nvmeibs	guids	Used for port filtering functionality.
nvmeibs	ib_port_prio	IB Port Priority (uint). Provides a mechanism to determine the

		preference of the mode in which to use NICs. Default is 0, lower is preferred.
nvmeibs	ignore_disks	Comma separated list of PCI IDs of NVMe drives to ignore.
nvmeibs	ignore_disks_serial	Comma separated list of Serial IDs of NVMe drives to ignore.
nvmeibs	ioka_timeout_sec	Controls the time to fail an IO channel due to keep alive failure. Default is 8 (seconds).
nvmeibs	local_skip_disk_access	Unsafe debug mode: Local client skip disk access. Default is off.
nvmeibs	max_client_rsrc	Maximum number of RDDA connections per client. Default is 128.
nvmeibs	max_completions	Maximum number of networking completions to handle per interrupt. Default is 64.
nvmeibs	max_local_nvmeqs	Maximum nvme queues for local operation. A value of 0 sets the actual maximum to the lower of the number of CPUs, drive queues, doorbells and MSI-X interrupts available. The default is 0.
nvmeibs	max_nic_srqs	Maximum number of shared receive queues to define per NIC. Default is 16.
nvmeibs	max_outstanding_cm_work_items	Max outstanding CM (connection manager) work items. Default is 8. Important for larger environments.
nvmeibs	max_req_size	Maximum size of client-target messages.
nvmeibs	min_local_nvmeqs	Minimum number of NVMe queues per drive to reserve for non-RDDA usage. Default is 1.
nvmeibs	mlx_rdda_enabled	Implements MLX_RDDA_ENABLED nvmesh.conf functionality.
nvmeibs	mostly_idle_ch	Use first shared CQ for mostly idle channels (bool). Default is false.
nvmeibs	nr_max_channels_per_path	The maximum No-RDDA RDMA channels per path (uint). Default is 4.
nvmeibs	nr_max_channels_per_path_tcp	The maximum No-RDDA TCP channels per path (uint). Default is 16.
nvmeibs	nr_max_wrs_per_req	Maximum number of WRs per No-RDDA channel request, used in response to a read request. For 0 (default), use system's default
nvmeibs	nr_post_rcv_on_send_comp	In percpu CQ and SRQ mode, post rcv buff on send comp of i-rsp. Default is false.

		Do not change without guidance from <a href="#">Excelero Technical Support</a> .
nvmeibs	nr_skip_disk_access	Unsafe debug mode: No-RDDA skip disk access. Default is off. Do not change without guidance from <a href="#">Excelero Technical Support</a> .
nvmeibs	nr_skip_rdma_write_back	Unsafe debug mode: No-RDDA skip rdma write-back in read operation. Default is off. Do not change without guidance from <a href="#">Excelero Technical Support</a> .
nvmeibs	nr_wq_set_cpu_affinity	Set CPU affinity of No-RDDA WQ (based on channel index). Default is false.
nvmeibs	num_warnings	The number of warnings the module has triggered. Contact support if this is not 0.
nvmeibs	nvmeibs_nordda_io_req_num	Number of IO requests per no-RDDA IO channel. Default is 96. More will increase throughput. Less will reduce memory consumption.
nvmeibs	nvme_number_offset	Offset for /dev/nvme%d device names. Default is 1000.
nvmeibs	ports	Used for port filtering functionality.
nvmeibs	qid_hint	Send data on a channel per the CPU id, mainly relevant for TCP. Default is false.
nvmeibs	roce_ipv4_only	Use IPv4 only for RoCE (Needed for CX-3). Default is false, deprecated.
nvmeibs	roce_port_prio	RoCE Port Priority (uint). Provides a mechanism to determine the preference of the mode in which to use NICs. Default is 10, lower is preferred.
nvmeibs	service_guid	Override cm_listen_id with this value.
nvmeibs	shared_rq_size	Networking shared receive queue (SRQ) size. Default is 32767.
nvmeibs	stamp_free_jrnl_entries	Stamp free journal entries, internal. Default is true.
nvmeibs	submit_wait_timeout	Timeout for NVMe admin operations such as format. Value in milliseconds.
nvmeibs	tcp_mode	TCP transport mode, 0 = RoCE only, 1 = TCP Only, 2 or greater = TCP and RoCE. If TCP is enabled, locks to all disks are done via CPU i.e. RPC or SIW (uint). Default is 0.
nvmeibs	tcp_port_prio	TCP Port Priority (uint). Provides a mechanism to determine the preference of the mode in which to use NICs. Default is 20, lower is preferred.

nvmeibs	tracer_debug_level	Control path tracing debug level. Default is 4.
nvmeibs	use_pcpu_cq	Use a per CPU shared completion queue (SCQ) and shared receive queue (SRQ). Default is false.

## SoftiWarp (SIW) Parameters, used for NVMesh over TCP

Module	Parameter Name	Description
siw	ack_signal_wr	Request responder to ack signaled writes (bool). Default is true.
siw	connect_non_block	Connect non-blocking (bool). Default is true.
siw	debug_level	Debug level, 1 for errors & events, 2 for more (int). Default is 1.
siw	iface_list	Interface list siw attaches to if present (array of characters). Default is "".
siw	loopback_enabled	enable_loopback (bool). Default is true.
siw	low_delay_tx	Run tight transmit thread loop if activated (bool). Default is true.
siw	low_delay_tx_cpu_set	bitmap of tx-cpus thread in tight loop (ulong). Default is all CPUs.
siw	mpa_crc_required	MPA CRC required (bool). Default is false.
siw	mpa_crc_strict	MPA CRC off enforced (bool). Default is true.
siw	notify_on_wq	Notify CQ on Workqueue (bool). Default is true.
siw	notify_on_wq_same_core	Notify CQ on Workqueue on the same core (bool). Default is false.
siw	panic_on_rx_err	Panic on RX Error (bool). Default is true.
siw	sock_buff_sz	Socket Buffers Size. Default is 64k.
siw	tcp_nodelay	Set TCP NODELAY (bool). Default is true.
siw	tcp_quickack	Set TCP QUICKACK (bool). Default is true.
siw	tx_cpu_list	List of CPUs siw TX thread shall be bound to (format: comma separated no spaces) (string). Default is none.
siw	tx_flags_from_upstream	Determines whether to take Tx flags from upstream version (bool). Default is false.
siw	tx_flags_use_eor	Determines whether to take Tx flags from upstream for use EOR (bool). Default is false.
siw	tx_thread_high_prio_bmp	A bitmap of CPU Tx Threads to set to high priority. Default is 0.
siw	tx_thread_same_cpu	Use same CPU for TX Thread. Default is true.
siw	tx_thread_scq_vector	Choose TX Thread based on SCQ Vector. Default is false.

siw	use_pbe_fixed_size	TBD.
siw	use_so_incoming_cpu	Set the RX CPU of socket to RCQ's comp-vector index (after connect/accept). Default is true.
siw	wait_rqe_delay_ms	Delay to wait on empty S(RQ) (in ms) (int). Default is 10.
siw	wait_rqe_max_retries	Number of retries on empty S(RQ) (int). Default is 10.
siw	zcopy_tx	Zero copy user data transmit if possible (bool). Default is true.
siw	zero_delay_tx	Run tight transmit thread loop always (bool). Default is true.

## Example

All examples run with root privilege.

To see the current value:

```
cat /sys/module/nvmeibc/parameters/max_ios_per_cpu
```

To change the value to 8 while the NVMesh client is running:

```
echo 8 > /sys/module/nvmeibc/parameters/max_ios_per_cpu
```

To change the value to 8 the next and subsequent times the client service is started/restarted:

```
echo "options nvmeibc max_ios_per_cpu=8" >> /etc/modprobe.d/nvmesh.conf
```

# 8. Storage Configuration

---

**NVMesh 2.5.2** enables the administrator to have fine-grained control over how logical volumes are provisioned. This is accomplished by defining and using *Target Classes*, *Drive Classes* and *Volume Provisioning Groups* (VPGs).

## 8.1. Target Classes

---

### Overview

*Target Classes* are logical groupings of target hosts. *Target Classes* are useful in grouping target hosts into distinct classes for limiting host and device selection during *Volume* creation. Example uses of *Target Classes* are to limit the use of certain target hosts by attributes such as their rack location, group ownership or redundancy level.

*Target Classes* are created by giving a name and description to groups of **Targets** by selecting specific **Targets** that have registered with a **Management**.

### Creating Target Classes

To create a new *Target Class*:

1. Click **Settings**, **Target Classes**.
2. Click the **+** to open a *Target Class* dialog box.
3. In the **Name** field, enter a name for the new *Target Class*.
4. (Optional) In the **Description** field, enter a description for the new *Target Class*.
5. (Optional) In the **Protection Domains** field, define *Protection Domains* to be associated with the **Targets** in this *Target Class*. Each *Protection Domain* is denoted using the following format, <protection domain scope: protection domain identifier>.
  - a. Multiple *Protection Domains* can be inserted. Use a comma to move from one *Protection Domain* to the next.
6. Multi-select from the **Targets** table.
7. Click **Add**.
8. Click **Save Changes** to complete the operation.

## Target Class ✕

**Name**

**Description**

**Protection Domains**

10 targets selected 1 - 10 out of 30 < > 10 ▾

10 ▲ Target ID	Drives	NICs	Version
<input checked="" type="checkbox"/> <input style="width: 40%; border: 1px solid #ccc;" type="text" value=".*-1[0-9].*"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="Search by Version"/>
<input checked="" type="checkbox"/> scale-10.excelero.com	2	2	v2.0.0-259-SIM
<input checked="" type="checkbox"/> scale-11.excelero.com	2	2	v2.0.0-259-SIM
<input checked="" type="checkbox"/> scale-12.excelero.com	2	2	v2.0.0-259-SIM
<input checked="" type="checkbox"/> scale-13.excelero.com	2	2	v2.0.0-259-SIM
<input checked="" type="checkbox"/> scale-14.excelero.com	2	2	v2.0.0-259-SIM
<input checked="" type="checkbox"/> scale-15.excelero.com	2	2	v2.0.0-259-SIM
<input checked="" type="checkbox"/> scale-16.excelero.com	2	2	v2.0.0-259-SIM
<input checked="" type="checkbox"/> scale-17.excelero.com	2	2	v2.0.0-259-SIM
<input checked="" type="checkbox"/> scale-18.excelero.com	2	2	v2.0.0-259-SIM
<input checked="" type="checkbox"/> scale-19.excelero.com	2	2	v2.0.0-259-SIM

## 8.2. Drive Classes

### Overview

*Drive Classes* are logical groupings of storage devices, e.g. SSDs. *Drive Classes* are useful in grouping *Drives* into distinct classes for use in device selection during *Volume* creation. Example uses of *Drive Classes* are to group certain types of *Drives* together by feature such as high performance, low write

endurance, size, etc. *Drives* can be grouped by business or social parameters such as purchase group, data type or project.

*Drive Classes* are created by selecting specific *Drives*.

## Creating Drive Classes

To create a new *Drive Class*:

1. Click **Settings, Drive Classes**.
2. Click the **+** to open a *Drive Class* dialog box.
3. In the **Name** field – enter a name for the new *Drive Class*.
4. (Optional) In the **Description** field – enter a description for the new *Drive Class*.
5. (Optional) In the **Protection Domains** field, define *Protection Domains* to be associated with the *Drives* in this *Drive Class*. Each *Protection Domain* is denoted using the following format, <protection domain scope: protection domain identifier>
  - a. Multiple *Protection Domains* can be inserted. Use a comma to move from one *Protection Domain* to the next.
6. Multi-select from the **Drives** table.
7. Click **Add**.
8. Click **Save Changes** to complete the operation.

Drive Class ✕

**Description**

**Protection Domains**

13 drives selected 1 - 10 out of 20 < > 10 ▾

13	▲ Target	SN	Vendor	Model	Block Size	Metadata Size	Health
<input checked="" type="checkbox"/>	scale-1[0-9].excelero	<input type="text" value="Search by Serial Nun"/>	<input type="text" value="Search by"/>	<input type="text" value="Search by Model"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input checked="" type="checkbox"/>	scale-10.excelero.com	RPTMNVOC0CU6.0	Intel	INTEL SSDPE2ME400G4	4KB	8B	✔
<input checked="" type="checkbox"/>	scale-10.excelero.com	RPTMNVOC0CU6.1	Intel	INTEL SSDPE2ME400G4	4KB	8B	✔
<input checked="" type="checkbox"/>	scale-11.excelero.com	D9QCPAXW876W.0	Intel	INTEL SSDPE2ME400G4	4KB	8B	✔
<input checked="" type="checkbox"/>	scale-11.excelero.com	D9QCPAXW876W.1	Intel	INTEL SSDPE2ME400G4	4KB	8B	✔
<input checked="" type="checkbox"/>	scale-12.excelero.com	PZ2U247HXMS4.0	Intel	INTEL SSDPE2ME400G4	4KB	8B	✔
<input checked="" type="checkbox"/>	scale-12.excelero.com	PZ2U247HXMS4.1	Intel	INTEL SSDPE2ME400G4	4KB	8B	✔
<input checked="" type="checkbox"/>	scale-13.excelero.com	5KN47FS2632Q.0	Intel	INTEL SSDPE2ME400G4	4KB	8B	✔
<input checked="" type="checkbox"/>	scale-13.excelero.com	5KN47FS2632Q.1	Intel	INTEL SSDPE2ME400G4	4KB	8B	✔
<input checked="" type="checkbox"/>	scale-14.excelero.com	G1M3W63CRWR6.0	Intel	INTEL SSDPE2ME400G4	4KB	8B	✔
<input checked="" type="checkbox"/>	scale-14.excelero.com	G1M3W63CRWR6.1	Intel	INTEL SSDPE2ME400G4	4KB	8B	✔

## 8.3. Volume Provisioning Groups

*Volume Provisioning Groups* (VPGs) are groupings of logical volume parameters. The parameters encompass [Logical Volume Types](#) as well as [Drive Classes](#) and [Target Classes](#), which potentially limit the **Targets** and **Drives** selected during volume creation. VPGs are later used in the volume provisioning process. When a *Drive* that is used for implementing a volume generated from a *Volume Provisioning Group* is evicted, the system will attempt to automatically allocate alternative space per the *Volume Provisioning Group* restrictions and automatically rebuild the volume. For volumes defined manually, this will not be done automatically.

VPGs comprises the following fields:

### Name

A name or identifier for this VPG.

### Description (Optional)

An informative description for this VPG.

## Volume Type

Volumes created with this VPG will be limited to the volume type specified.

## Target Classes (Optional)

Volumes created with this VPG will have their device selections limited to **Targets** that are members of the specified *Target Classes*.

## Drive Classes (Optional)

Volumes created with this VPG will have their drive selections limited to *Drives* that are members of the specified *Drive Classes*.

## Protection Domains (Optional)

*Protection Domains* by which to separate data copies to ensure high availability across this protection domain.

## VPG Reserve Space

Defines the capacity of space to be reserved for volumes that are created utilizing this VPG.

 Be sure to check the unit of measure (GB, TB, etc.)

Volume Provisioning Group
✕

**Name**

**Description**

**Volume Type**

**Stripe Width:**

**Target Classes**

**Drive Classes**

**Protection Domain**

**VPG Reserve Space**

30TB/59.39TB

Mirror #1

*Total drives: 158*

For convenience, **NVMesh 2.5.2** includes some default VPGs that make creating volumes easier. The default VPGs are:

### DEFAULT\_CONCATENATED\_VPG

This VPG is used to generate concatenated volumes.

### DEFAULT\_RAID\_0\_VPG

This VPG is used to generate RAID-0 volumes.

## DEFAULT\_RAID\_1\_VPG

This VPG is used to generate RAID-1 volumes.

## DEFAULT\_RAID\_10\_VPG

This VPG is used to generate RAID-10 volumes.

## DEFAULT\_EC\_DUAL\_TARGET\_REDUNDANCY\_VPG

This VPG is used to generate erasure coded volumes with dual target node redundancy, i.e. the volume will continue to provide service even after two nodes on which its data is located are unavailable. This implies that drive space from at least 10 *Drives* on 10 **Targets** will be needed for an 8+2 erasure coded volume.

## DEFAULT\_EC\_SINGLE\_TARGET\_REDUNDANCY\_VPG

This VPG is used to generate erasure coded volumes with single target node redundancy, i.e. the volume will continue to provide service even after a node on which its data is located is unavailable. This implies that drive space from at least 10 *Drives* on 5 **Targets** will be needed for an 8+2 erasure coded volume.

# 8.4. Protection Domains

---

*Protection Domains* are a mechanism to assist in ensuring data availability is aligned to specific data center protection requirements.

Out of the box, **NVMesh 2.5.2** separates mirrored copies of data or erasure coded elements to *Drives* on different **Targets**. However, all **Targets** and *Drives* within the same *Zone* are considered equivalent.

*Protection Domains* enable enhancing the separation layout with user-defined criteria.

For instance, it may be prudent to separate data for a crucial volume that should always be available to 2 separate racks. Even if there is a total rack failure the volume's data will still be accessible. Other data separation criteria could be defined for different fault or availability zones such as power supplies, fire suppression systems, data center rows or upgrade zones.

To utilize the *Protection Domain* capabilities, it is necessary first to define the *Protection Domain* in which resources are located. Then, during the volume creation process, the system can be guided on how to separate the data elements. For instance, for rack separation, each **Target** would be labeled with its rack identifier. Then, upon volume creation, "rack" would be set as a separation *Protection Domain*.

Instead of assigning *Protection Domain* information directly to **Targets** and *Drives*, this is done by grouping them in *Target Classes* and *Drive Classes*, respectively, and then associating one or more *Protection Domains* with them. For a specific *Target Class* or *Drive Class* the information is provided using a list of key-value pairs. The key is the *Protection Domain*'s type and the value is the specific instance. For example, the *Protection Domain* type could be "rack" and the specific instance value could be "C-33". In this case, the key-value pair list would be {"`rack`": "C-33"}. These values can be inserted for *Target Classes* and

*Drive Classes* via the **Management** GUI or using the RESTful API. At volume creation, only the *Protection Domain* keys should be provided. In this example, it would be “rack”. This functionality enables fine-tuned allocation. For instance, it is possible to create a volume limited to 2 racks by choosing the *Target Classes* tagged with “rack” keys “C-32” and “C-33” to ensure data is protected by separated it between these two racks.

## 8.5. Read-only Access

---

By default, block devices will be read-write or read-only based on the mode in which they were opened by applications, typically the overlying file system.

When an application opens a block device in read-only mode, flags will be sent to the block device as part of the open call, which will inform the **Client** that it should consider the block device to be read-only. Sometimes, the operating system will still attempt to write data to this volume. The default behavior for **NVMesh 2.5.2** is to prevent such writes. This behavior can be augmented. For more details, contact [Excelero Technical Support](#).

This behavior enables a common usage pattern of using a local file system for sharing quasi-static data efficiently to a large number of readers where the file system is mounted read-only. In this pattern, the volume is attached to all readers and mounted in a read-only mode.

For data update, the following steps are taken:

1. The clients unmount the file system.
2. The update is done from a single client with a read-write mount. The updater then unmounts the file system.
3. The client remount the file system in read-only mode.

For example, if a volume is mounted as read-only, all write operations to the volume will be failed, including trim commands:

```
[root@nvme31 17:50:55 nvmesh]# mount -o ro /dev/nvmesh/tlro /mnt/tlro
[root@nvme31 17:51:00 nvmesh]# fstrim --verbose /mnt/tlro
fstrim: /mnt/tlro: FITRIM ioctl failed: Input/output error
[root@nvme31 17:51:02 nvmesh]# mount /dev/nvmesh/tlro /mnt/tlro
[root@nvme31 17:51:17 nvmesh]# fstrim --verbose /mnt/tlro
/mnt/tl: 920.3 MiB (964972544 bytes) trimmed
```

## 8.6. Command Line Client Operations

---

**NVMesh 2.5.2** supports command line operations on the **Clients** for volume manipulation and for controlling multi-instance functionality.

The following operations are supported:

- [Attaching volumes](#)
- [Detaching volumes](#)
- [Checking volume status](#)
- [Managing multiple \*Client\* instances](#)

## 8.6.1. Attaching Volumes

After volumes have been provisioned using **Management Servers**, they can be attached from the command line on a **Client** using `nvmesh_attach_volumes` as follows.

```
usage: nvmesh_attach_volumes [-h] [--debug] [-w] [-d] [-v] [-u] [-c]
                             [-n NAMED] [-m MC] [-t TIMEOUT] [-i MCI]
                             [-a ACCESS] [-p] [-r READ_AHEAD_KB]
                             [--sub-block-io-allowed]
                             [--management_cluster MANAGEMENT_CLUSTER]
                             [--management_http_port MANAGEMENT_HTTP_PORT]
                             [--management_protocol {https,http}]
                             [--configured_nics CONFIGURED_NICS]
                             [--rdda | --no-rdda]
                             [volumes [volumes ...]]

positional arguments:
  volumes                volumes to be attached

optional arguments:
  -h, --help            show this help message and exit
  --debug              print attach debug information
  -w, --wait_for_attach
                        wait for attach to finish
  -d, --hidden         attach volume as hidden
  -v, --recover        attach volume as recoverer
  -u, --upgrade        adopt existing volume
  -c, --no_cancel      do not cancel attach request on timeout
  -n NAMED, --named NAMED
                        create an alias for volume use [auto] for a random
                        choice
  -m MC, --mc MC       multi client instance
  -t TIMEOUT, --timeout TIMEOUT
  -i MCI, --mci MCI    multi client instance string
  -a ACCESS, --access ACCESS
                        Volume access mode: EXCLUSIVE_READ_WRITE,
```

```

        SHARED_READ_ONLY, SHARED_READ_WRITE
-p, --preempt          Use preempt for applying access mode
-r READ_AHEAD_KB, --read-ahead-kb READ_AHEAD_KB
                        read ahead kb value to be used for the volumes given
--sub-block-io-allowed
                        Allow kernel (512B) sector aligned IO operations
--management_cluster MANAGEMENT_CLUSTER
                        The Management cluster that manages the volume. The
                        string should be in the following format: "<server
                        name or IP>:<port>,<server name or IP>:<port>,..."
--management_http_port MANAGEMENT_HTTP_PORT
                        The management http server port, the default is 4000.
--management_protocol {https,http}
                        The protocol of the management cluster, the default is
                        https.
--configured_nics CONFIGURED_NICS
                        Define the NICs for the client instance. The string
                        should be in the following format:"<interface
                        name/transport type;interface name/transport
                        type;...>". Transport type options: TCP/RDMA.
--rdda                optional: enable RDDA for instance
--no-rdda             optional: disable RDDA for instance

```

After running the attach command, any volumes available to the **Client**, out of those specified, become available as block devices with the following path:

```
/dev/nvmesh/<vol_name>
```

Although entries in `/dev/nvmesh/<vol_name>` look and behave like regular block devices, they are special and controlled by **NVMesh 2.5.2**. Manual changes to the directory `/dev/nvmesh` and its contents can lead to instability in the system. This directory and its contents should not be modified by the end-user or system administrator.

By default, the `nvmesh_attach_volumes` command will return as soon as the attachment operations have completed and appear in `/dev/nvmesh/<vol_name>`. It is important to note that a successfully attached volume may not be immediately ready for IO. If the expectation is that when the attach command completes, IO can immediately proceed, then the `nvmesh_attach_volumes` command should be run with the `-w / --wait_for_attach` option. By specifying this option, the `nvmesh_attach_volumes` command will not return until the attach is successful and IO is enabled for all specified volumes and may take an indeterminate amount of time. Without specifying this option, immediate IO operations may subsequently time out after 30 seconds, indicating that IO had never been enabled for this volume.

\* Note: Once IO has been enabled, the timeout for IO operations is practically infinite.

To check a volume's status to determine if IO is enabled, see [Checking Volume Status](#).

If the **Client** is unable to communicate with one of the **Management Servers** and does not have a cached copy of the volume definition, the `nvmesh_attach_volumes` command will fail. Adding the `-c / --no_cancel` will make the command retry indefinitely.

Once a logical volume is attached, it will automatically attach at service restart unless auto-attach has been disabled for the **Client**.

The `--debug` option is useful when interacting with [Excelero Technical Support](#) if required.

The `-i / --mci` option can be used to specify to which **Client** instance to attach the volume. See [Multiple Client Instances](#) for more information.

The `-a / --access` option can be used to specify the access mode with which to attach the volume. See [Access Modes](#) for more information.

The `-p / --preempt` option is used to preempt another client attached to the volume to enable this client to attach with the requested access mode.

The `-n / --named` option is used to generate an alias or alternative name with which to access the volume. The same volume can be attached with multiple aliases or alternative names. To complete detaching the volume, it should be detached under all its names. This functionality is useful for running multiple jobs independently that require access to the same volume.

The `-d / --hidden` option is used for internal recovery operations and should not be used for regular storage access.

The `--sub-block-io-allowed` option is used to attach a volume with 512 byte block size emulation, see [512-byte Block Size Emulation](#) for more details.

\* There are some additional advanced options not described here that are for functionality that is not generally available in **NVMesh 2.5.2**.

## 8.6.2. Detaching Volumes

\* Before detaching a volume, it is important to make sure it does not have a mounted file system utilizing it.

To detach a volume from the command line, use:

```
/usr/bin/nvmesh_detach_volumes [-f / --force] [-s / --client_shutdown] [--debug] [-a / --all] <vol_name1 [vol_name2] ...>
```

Once a volume is explicitly detached, it will no longer automatically attach at service restart.

With normal command invocation, without parameters, the `nvmesh_detach_volumes` command will refuse to detach any volume that is in use, i.e. its block device `/dev/nvmesh/<vol_name>` is open, and the command will return a failure error code.

If the `-f / --force` parameter is used, the `nvmesh_detach_volumes` command will detach the volume even if it is in use. New IO to this device will fail immediately and the block device `/dev/nvmesh/<vol_name>` will be removed. This is roughly equivalent to pulling a local drive out from a system without warning and can result in data loss if the host has any cached, uncommitted writes.

The `-s / --client_shutdown` parameter is used during *client* service shutdown and should not be run from the command line.

The `--debug` option is useful when interacting with [Excelero Technical Support](#) if required.

The `-a / -all` parameter directs to detach all currently attached volumes. In this case, there is no need to specify volume names.

\* Note: If there are open file handles to any block device, the *Client* service will not stop until they are closed. The service shutdown operation will hang in such a case. This applies also for volumes that were stopped with the `--force` parameter.

\* There are some additional advanced options not described here that are for functionality that is not generally available in *NVMesh 2.5.2*.

## 8.6.3. Checking Volume Status

To verify a volume's size and to see other device attributes, use:

```
fdisk -l /dev/nvmesh/<vol_name>
```

Detailed information on the status of an *NVMesh 2.5.2* block device is available in `/proc/nvmeibc/volumes/<vol_name>/status`. The information is best viewed in a wide window.

For example:

```
[root@nvme241 16:54:09 ~]$ cat /proc/nvmeibc/volumes/EcVol1_8_2/status
Name=EcVol1_8_2, UUID=2198df50-471a-11ea-be41-bd5b0ac74247, size=13427712[block
s], 52452[Mb] Sector Size=4096[bytes], ptr=ffff94d629e5d000, type=visible
```

```

Device status: Attaching, Live, no IO, partial Error (debug:0x1a1, 0)
IO is currently disabled for 0 [sec].
Raid Type: RAID-6
Topology Debug: i=[16975..16974], ver=1, io_perm=-1 nr=0 ns=0[blks]
Chunk #0: Stripe{Size=256, Width=1} Slice{ 8+2} Vol Blocks [0..13427711]

```

t	Stripe	Slice	Status	Disk NVMe ID	LBA Star
	0	0	Online	S3HCNX0K500635.1	150928
0	3187743		nvme244.excelero.com	[a=1 p=0 acm=RW	sy=1 lm(O
0,C9,C8)	r1v=0x107	lid=0x410 a	uid=219a8d00]		
	0	1	Online	S3HCNX0K701052.1	150928
0	3187743		nvme243.excelero.com	[a=1 p=0 acm=RW	sy=1 lm(O
1,C0,C9)	r1v=0x107	lid=0x410 a	uid=219ab410]		
	0	2	Offline, stops IO	S3HCNX0K500691.1	150928
0	3187743		Unknown	[a=0 p=3 acm=RW	sy=1 lm(O
2,C1,C0)	r1v=0x107	lid=0x410 a	uid=219b0230]		
	0	3	Online	S3HCNX0K600182.1	150928
0	3187743		nvme244.excelero.com	[a=1 p=0 acm=RW	sy=1 lm(O
3,C2,C1)	r1v=0x107	lid=0x410 a	uid=219b0231]		
	0	4	Online	S3HCNX0JC02108.1	150928
0	3187743		nvme243.excelero.com	[a=1 p=0 acm=RW	sy=1 lm(O
4,C3,C2)	r1v=0x107	lid=0x410 a	uid=219b2940]		
	0	5	Offline, stops IO	S3HCNX0K500629.1	150928
0	3187743		Unknown	[a=0 p=3 acm=RW	sy=1 lm(O
5,C4,C3)	r1v=0x107	lid=0x410 a	uid=219b5050]		
	0	6	Online	S3HCNX0JC02111.1	150928
0	3187743		nvme243.excelero.com	[a=1 p=0 acm=RW	sy=1 lm(O
6,C5,C4)	r1v=0x107	lid=0x410 a	uid=219b7760]		
	0	7	Offline, stops IO	S3HCNX0K600183.1	150928
0	3187743		Unknown	[a=0 p=3 acm=RW	sy=1 lm(O
7,C6,C5)	r1v=0x107	lid=0x410 a	uid=219b9e70]		
	0	8	Offline, stops IO	S3HCNX0JC02157.1	150928
0	3187743		Unknown	[a=0 p=3 acm=RW	sy=1 lm(O
8,C7,C6)	r1v=0x107	lid=0x410 a	uid=219bc580]		
	0	9	Offline, stops IO	S3HCNX0K600272.1	150928
0	3187743		Unknown	[a=0 p=3 acm=RW	sy=1 lm(O
9,C8,C7)	r1v=0x107	lid=0x410 a	uid=219bec90]		

```

Enforce Read Only: Y, Retry Timeout: 30[sec]
Sync Stats: r=0/p=0, OK:f=0/p=0 Done:t=0/m=0, longest=0[msec]
Failed IO: crit=0, detach=0, ignore=0, other=0 (resub: in=2, out=0, tout=2) {su
s_thresh=7, n_binfo_err=0}
Mgmt alerts: arm[cur=io-disabled stable=io-disabled] last_armed] {at=4294724, uw2
ro=0}[sec] stats={sent=0, longest=0[sec]}, conf={stab=10, unpW=1844674407370955
1, detStu=0}[sec]

```

```
Slow IO stats: num_over_retry=0, lock_id=0x0 blkset{slba=0x0,dlba=0x0}, se
g=(0,0,0)
HOSTNAME: 'nvme241.excelero.com', PROC: 'dev_status', TAKEN_ON: '05/02/2020 14:5
4:22 (UTC)
```

This output provides the following information:

- In the first line of output, the volume name and size. Volume type appears in the fourth line.
- In the second line of output, basic status, for instance in the example above, the volume is in the process of initial attach, which means that it has not been enabled for I/O yet.
- The third line specifies whether IO is currently enabled for this volume. The **Client** will not reject IO, instead waiting for IO to become enabled before returning an error. However, this will give an indication on the expected latency or immediacy for IO operations.
- After the 5th line, there is a description of the drive segments comprising the volume and their current state.
- Following this an indication of whether the volume enforces read-only mode behavior, see [Read-only Access](#), and the current retry timeout for IO. This is the value that begins at 30 and is raised upon the first successful I/O operation.
- The output contains lots of additional information that may be useful for [Excelero Technical Support](#), if needed.



The contents and format of this file are fluid and may change without notice. Therefore, it is not recommended to build parsers for it. For parsing, the status.json file is more tuned for machine-reading, as it is in a JSON format. Its contents may also change without any notice.

The client\_processes file co-located with the status file can provide information to assist in detaching a volume that is busy.

There are lots of other files with mostly debug information under /proc/nvmeibc. Content format may change without any notice.

## IO Disabled

IO is sometimes disabled for a specific client for a specific volume. This can be seen in the **Management Servers'** GUI, see [Volume State](#) or by looking at the state field in the volume's status file, as described above. The root cause may be a missing *Drive*, a **Target** failure or a network connectivity failure.

IO may also be in a suspended state. This will happen if a potential corruption was identified triggered by an internal contradiction detected in the software layers. For instance, if the dirty bits collected are not aligned with the number of drives required for erasure coded volumes, IO will be suspended. Another example is unidentified NVMe errors, which may lead to suspension. To relieve the suspended state on the volume, [detach](#) it and then [attach](#) it. Another option is to run the following command, as root: `echo "#vol_name|v`

olume\_suspend=0" > /proc/nvmeibc/cli/cli, note that vol\_name is a placeholder for the actual volume name.

## 8.7. Drive Segment Zeroing

---

During initial formatting of a drive the system trims the entire drive. By default, the system will use the NVMe trim command to achieve this. This behavior can be altered to actually write zeroes if needed either using a special NVMe command to this extent or via writing zeroes directly.

In addition, when a volume is deleted, it is assumed that blocks may contain previously-written data, and therefore these blocks must be zeroed out before reuse. This zeroing starts immediately after the volume is deleted. As a result, **Client** reads from the volume will not fetch old data from the drives.

The zeroing behavior is governed by three parameters set directly on **Targets** using **toma\_rpc**.

To alter behavior run commands such as in the following example that sets the current default values, which are to TRIM without verifying that the blocks are later read as zeroes.

```
/opt/nvmesh/common-repo/tools/toma_rpc disk-models default is_using_nvme_trim_bef  
ore_zero on  
/opt/nvmesh/common-repo/tools/toma_rpc disk-models default is_zeroing_using_tes  
t_and_write off  
/opt/nvmesh/common-repo/tools/toma_rpc disk-models default is_zeroing_mandatory o  
ff
```

- **is\_using\_nvme\_trim\_before\_zero** governs whether to use TRIM operations as a means of zeroing.
- **is\_zeroing\_using\_test\_and\_write** governs whether to read and write zeros as a means of zeroing.
- **is\_zeroing\_mandatory** can be used to stop all zeroing.

## 8.8. Volume Rebuild Prioritization

---

In the volume definition dialog, there are controls for managing rebuild prioritization. They define the relative load to place on rebuilding a volume, including network, disk and compute load versus serving standard I/O. Using a lower value reduces the load, increasing IO/s, but may cause rebuilds to take much more time. Rebuild priority is defined per volume and is applicable to all rebuild functions on that volume.

The supported values are between 1 and 10, in units of 10%. 10 means maximal rebuild speed (100%), while 1 means 10%, which is the lowest speed available.

A special value of 0 means that the load is defined on the **Targets** individually. On each target, the priority or speed of rebuild can be configured via "ioctls".

The granularity of configuration is more precise and allows to set a specific rebuild priority value for each

function or type of rebuild. Contact [Excelero Technical Support](#) for fine-grained control instructions.

The rebuild priority parameter does not affect which volumes are rebuilt first.

There are other TOMA-specific parameters that control the number of concurrent volume rebuilds per target.

# 9. Management Configuration

---

## 9.1. Management Server Options

---

**Management Servers'** options reside in the configuration file at this path, `/etc/opt/NVMesh/management.js.conf`

This file is directly read by the **Management**. The first line and last line in the file should not be modified. They file format should remain as follows.

```
var config = {};  
...  
module.exports = config;
```

The following subsections list the option parameters with their descriptions, defaults and possible values.

### 9.1.1. General Options

---

#### **config.autoLogoutThreshold**

**Description:**

The timeout of the GUI and API access, in milliseconds. After the timeout expires, the GUI and API will automatically logout all logged in users.

**Default Value:**

`1000 * 60 * 60`, 60 minutes.

**Possible Values:**

`1000 * 60` to `1000 * 60 * 60 * 24 * 30`, from 1 minute to 1 month.

#### **config.forcelP**

**Description:**

Force an IP address to use for accepting and sending management traffic.

**Default Value:**

None.

**Possible Values:**

IP address.

**config.keepaliveGracePeriod****Description:**

The grace period, in milliseconds, since the last message for every component. When the grace period is over, *Management Servers* will consider the component as timed out.

**Default Value:**

1000 \* 60 \* 5.

**Possible Values:**

1000 \* 30 to 1000 \* 60 \* 30, from 30 seconds to 30 minutes.

**config.MAX\_JSON\_SIZE****Description:**

The size of the largest JSON message supported by the Management Server. Do not modify this setting unless explicitly authorized by Excelero.

**Default Value:**

2mb

**Possible Values:**

2mb

**config.port****Description:**

The TCP port the management server should listen on.

**Default Value:**

4000

**Possible Values:**

A valid TCP port number

## config.RESERVED\_BLOCKS

### Description:

The percentage of reserved blocks at the start of a managed NVMe device. Do not modify this setting unless explicitly authorized by Excelero.

### Default Value:

0.5

### Possible Values:

0.5

## config.websocket.\*

### Description:

Web socket parameters.

It is not recommended to modify these without direction or guidance from [Excelero Technical Support](#).

## config.webSocketServerPort

### Description:

The TCP port number to be used by management for dynamic updates from clients and targets.

### Default Value:

4001

### Possible Values:

A valid TCP port number that is not the same as **config.port**.

## config.statisticsWebSocketServerPort

### Description:

The TCP port number to be used by management for collecting statistics from clients and targets.

### Default Value:

4002

**Possible Values:**

A valid TCP port number that is not the same as **config.port** or **config.webSocketServerPort**.

**config.compatibilityMode****Description:**

Determines whether to run with ldlinux and shared libraries (for older Linux distribution compatibility).

**Default Value:**

false

**Possible Values:**

true or false (Generally, this should not be changed).

**config.enableLegacyFormatting****Description:**

Determines whether to allow legacy formatting, using 4k blocks, on metadata supported drives via the RESTful API.

**Default Value:**

false

**Possible Values:**

true or false.

**config.server.\*****Description:**

General Node.js server parameters.

It is not recommended to modify these without direction or guidance from [Excelero Technical Support](#).

**config.enableDistributedRAID****Description:**

Determines whether to allow creation of erasure coding volumes.

**Default Value:**

true

**Possible Values:**

true or false.

**config.enableFunctionExecutionTimers****Description:**

Used for debugging

**Default Value:**

false

**Possible Values:**

true or false.

**config.enableZones****Description:**

See the section on [Zones](#). This general switch enables the zoning functionality.

**Default Value:**

false

**Possible Values:**

true or false.

## 9.1.2. SSL Configuration Options

---

**config.useSSL****Description:**

Determines whether or not HTTP communication to the management server is encrypted via SSL.

**Default Value:**

false

**Possible Values:**

true or false

**config.cert****Description:**

Path to the file containing the SSL certificate to be used for HTTP encryption.

**Default Value:**

'cert/server.crt'

**Possible Values:**

A valid path to the stored certificate

**config.key****Description:**

Path to the file containing the key file for the certificate to be used for HTTP encryption.

**Default Value:**

'cert/server.key'

**Possible Values:**

A valid path to the stored key file.

## 9.1.3. MongoDB Connection Options

---

- The following options are described for `config.mongoConnection`, which stores the main *NVMesh 2.5.2* data.
- A second MongoDB database is used to store the statistics collected. This database can be organized differently per the performance and availability requirements. The same options apply for connecting to this database from `config.statisticsMongoConnection`.
- A third MongoDB database is used to store system metadata that describes the cluster layout. The same options apply for connecting to this database from `config.nvmeshMetadataMongoConnection`.

## config.mongoConnection.hosts

### Description:

The URI used to connect to the MongoDB server(s) containing the management database.

### Default Value:

```
hosts: 'localhost:27017'
```

### Possible Values:

A list of valid MongoDB servers utilizing hostnames, port numbers and the database name. For example, with three-way replication, this option value might look like:

```
hosts: 'host1:27017,host2:27017,host3:27017'
```

## config.mongoConnection.options.replicaSetName

### Description:

For management database redundancy, this setting is used to define the MongoDB Replica Set name. This value must be the same on all hosts.

### Default Value:

"" (empty)

### Possible Values:

An arbitrary text value. `replicaSetName: rs0` for example.

## config.mongoConnection.auth.username

### Description:

This is the username to be used for management database access control and authentication. Leave undefined in case no database access control is employed.

### Default Value:

"" (empty)

### Possible Values:

An arbitrary text value. `username: johndoe` for example.

## config.mongoConnection.auth.password

### Description:

This is the password to be used for management database access control and authentication. Leave undefined in case no database access control is employed.

### Default Value:

"" (empty)

### Possible Values:

An arbitrary text value. `password: NvmeshIsGr8` for example.

## config.mongoConnection.auth.authenticationDatabase

### Description:

This is the mongoDB administration database to be used for management database access control and authentication. Leave undefined in case no database access control is employed.

### Default Value:

"" (empty)

### Possible Values:

An arbitrary text value. `authenticationDatabase: admin` for example.

## config.mongoConnectOptions.server.socketOptions

### Description:

Settings defining the connection handling from the Management Server to MongoDB.

**!** It is not recommended to modify these settings without direction or guidance from [Excelero Technical Support](#).

## keepAlive

### Description:

If set to 1 use a keep-alive mechanism to keep the connection to MongoDB alive.

**Default Value:**

1

**Possible Values:**

0 or 1.

**connectTimeoutMS****Description:**

Connection timeout in milliseconds.

**Default Value:**

60000, 30 seconds.

**Possible Values:**

1000 to 300000, 1 second to 5 minutes.

**socketTimeoutMS****Description:**

Socket timeout in milliseconds.

**Default Value:**

60000, 60 seconds.

**Possible Values:**

1000 to 300000, 1 second to 5 minutes.

## 9.1.4. Log Level Options

---



The logging options below can also be configured from the *Management* GUI under the *General* subsection of the *Settings* section in the *Logging* settings area.

## config.loggingLevel

**Description:**

The logging level of the Management Server.

**Default Value:**

INFO

**Possible Values:**

DEBUG, INFO, WARNING, ERROR or VERBOSE. Do not modify this setting unless explicitly authorized by Excelero.

## config.debugComponents

**Description:**

Enable or disable the debug logging of each component of the Management Server.

## lock

**Description:**

Enable or disable the logging of management locking.

**Default Value:**

false

**Possible Values:**

true, false.

## events

**Description:**

Enable or disable the logging of management events.

**Default Value:**

false

**Possible Values:**

true, false.

**counters****Description:**

Enable or disable the logging of counters.

**Default Value:**

false

**Possible Values:**

true, false.

**client****Description:**

Enable or disable the logging of client.

**Default Value:**

false

**Possible Values:**

true, false.

**statistics****Description:**

Enable or disable the logging of statistics.

**Default Value:**

false

**Possible Values:**

true, false.

## diskSegments

**Description:**

Enable or disable the logging of *diskSegments*, an internal name for segments of a drive allocated to a volume.

**Default Value:**

false

**Possible Values:**

true, false.

## perf.updatePRaidStatus

**Description:**

Enable or disable the logging of *perf.updatePRaidStatus*, an internal name for the methods related to updating volumes statuses.

**Default Value:**

false

**Possible Values:**

true, false.

## config.daysBeforeLogEntryExpires

**Description:**

The number of days the management logs are kept before rotation.

**Default Value:**

30

**Possible Values:**

1 to 365.

## 9.1.5. SMTP Options

---

### config.SMTP

**Description:**

Various configuration options specifying which mail server to use for outbound messages.

#### host

**Description:**

The hostname or IP address of the mail server.

**Default Value:**

```
'localhost'
```

**Possible Values:**

A valid hostname or IP address.

#### port

**Description:**

The TCP port to be used on the specified mail server.

**Default Value:**

```
25
```

**Possible Values:**

A valid TCP port number.

#### secure

**Description:**

Determines where or not SMTP communication is encrypted.

**Default Value:**

```
false
```

**Possible Values:**

true or false

**authRequired****Description:**

Determines whether authentication is required for SMTP communication.

**Default Value:**

false

**Possible Values:**

true or false

**username****Description:**

If authRequired is true, utilize this username to authenticate to the SMTP server.

**Default Value:**

' ' (empty)

**Possible Values:**

User account name to be used for SMTP server account authentication.

**password****Description:**

If authRequired is true, password to be used for SMTP server account authentication.

**Default Value:**

' ' (empty)

**Possible Values:**

Plaintext password to be used for SMTP server account authentication.

## useDefault

**Description:**

Determines whether to assume Gmail SMTP server behavior.

**Default Value:**

true

**Possible Values:**

true or false

## config.exceleroEmail

**Description:**

The email address to send “phone home” statistics to, at Excelero.

**Default Value:**

```
'support+CustomerName@excelero.com'
```

**Possible Values:**

A valid email address, preferably the default value.

## config.sendStatsInterval

**Description:**

The interval of time passing after which the “phone home” statistics should be sent to **config.exceleroEmail**.

**Default Value:**

1w

**Possible Values:**

Minimal value is 1 hour. A numerical value will be in weeks. One can use the format `<N>w <M>d` to mean N weeks and M days.

## 9.1.6. Backup Options

---

### config.Backup

**Description:**

Various configuration options controlling aspects of the automatic backup of the management database.

### backupPath

**Description:**

The directory path where backup should be written.

**Default Value:**

```
' /var/opt/NVMesh/backups '
```

**Possible Values:**

A valid directory path name writable by the excelero user id.

### hourlyBackupInterval

**Description:**

How often the database should be backed up.

**Default Value:**

```
1
```

**Possible Values:**

An integer number of hours.

### dailyBackupTime

**Description:**

Time of the day that should be considered a daily backup.

**Default Value:**

```
"00:00" (Midnight)
```

**Possible Values:**

24-hour time value

## hourlyRotationThreshold

**Description:**

The number of hourly backups to keep before the oldest is rotated out (deleted).

**Default Value:**

36, the last 1.5 days.

**Possible Values:**

An integer value

## dailyRotationThreshold

**Description:**

The number of daily backups to keep before the oldest is rotated out (deleted).

**Default Value:**

30

**Possible Values:**

An integer value



The management backup is saved as an archived Mongo database dump. To restore it, run `mongorestore --gzip --archive=<backup_file>.tar.gz`. For more information on the `mongorestore` command, please refer to [MongoDB user guide](#).

## 9.1.7. Statistics Options

---

### config.cacheUpdateInterval

**Description:**

Frequency of an internal fail-safe mechanism to prevent statistics from getting out of sync (in milliseconds).

**Default Value:**

1000 \* 60

**Possible Values:**

1000 \* 60

## config.collectStatistics

**Description:**

General switch for enabling or disabling statistics collection.

**Default Value:**

true

**Possible Values:**

true or false.

## config.statisticsCores

**Description:**

The number of CPU cores to be used by the statistics processes.

**Default Value:**

5

**Possible Values:**

1 to the number of cores in the server.

## config.statisticsPorts

**Description:**

Comma-separate list of TCP ports to be used by statistics processes.

**Default Value:**

Ports 4002 and onwards with one port for every core use, see `config.statisticsCores`.

**Possible Values:**

[4002, 4003, 4004, 4005, 4006].

**config.requestStatsInterval****Description:**

The frequency of statistics updates from the node machines to the management server (in milliseconds).

**Default Value:**

8000

**config.fullClientReportInterval****Description:**

The frequency at which the management does a full sync with clients on all attachments, in milliseconds. It is not recommended to change this value without direction or guidance from [Excelero Technical Support](#)

**Default Value:**

300000

**Possible Values:**

30000 to 3600000, 30 seconds to hour.

**config.rollupPolicy****Description:**

The intervals at which the collected statistics data will be collapsed. There are four interval sets.

**samplingRate****Description:**

The rate in which data is sampled from clients and targets by the Management Server. Use `s` to denote seconds, `min` to denote minutes, `hr` to denote hours and `d` to denote days.

**Default Value:**

1s

**Possible Values:**

1s to 1hr.

## samplingDuration

**Description:**

The duration of the sampling interval. During this time window, data is sampled from clients and targets by the Management Server at the specified **samplingRate**. Use `s` to denote seconds, `min` to denote minutes, `hr` to denote hours and `d` to denote days.

**Default Value:**

2min

**Possible Values:**

1s to 1hr.

## samplingRate

**Description:**

The rate in which data is sampled from clients and targets by the Management Server. Use `s` to denote seconds, `min` to denote minutes, `hr` to denote hours and `d` to denote days.

**Default Value:**

30s

**Possible Values:**

1s to 1hr.

## samplingDuration

**Description:**

The duration of the sampling interval. During this time window, data is sampled from clients and targets by the Management Server at the specified **samplingRate**. Use `s` to denote seconds, `min` to denote minutes, `hr` to denote hours and `d` to denote days.

**Default Value:**

15min

**Possible Values:**

1s to 1hr.

## samplingRate

**Description:**

The rate in which data is sampled from clients and targets by the Management Server. Use `s` to denote seconds, `min` to denote minutes, `hr` to denote hours and `d` to denote days.

**Default Value:**

1min

**Possible Values:**

1s to 1hr.

## samplingDuration

**Description:**

The duration of the sampling interval. During this time window, data is sampled from clients and targets by the Management Server at the specified **samplingRate**. Use `s` to denote seconds, `min` to denote minutes, `hr` to denote hours and `d` to denote days.

**Default Value:**

1hr

**Possible Values:**

1s to 1hr.

## samplingRate

**Description:**

The rate of data sampling from clients and targets by the Management Server. Use `s` for seconds, `min` for minutes, `hr` for hours and `d` for days.

**Default Value:**

1hr

**Possible Values:**

1s to 1hr.

**samplingDuration****Description:**

During this duration, data is sampled from clients and targets by the Management Server at the specified `samplingRate`. Use s for seconds, min for minutes, hr for hours and d for days.

**Default Value:**

1d

**Possible Values:**

1s to 1hr.

## 9.2. Management Scalability

---

The **Management Servers** utilize NodeJS for front-end HTTP/HTTPS and MongoDB as a backend database. To achieve high availability, install MongoDB in a replica set configuration, with a minimum of 3 replicas. In addition, a minimum of 2 instances of NodeJS should be configured as well. In practice, the suggested high availability (HA) configuration is to have 3 redundant **Management Servers**.

In a MongoDB replica set configuration, data written into any one instance is replicated to the others using asynchronous commits for eventual consistency.

### 9.2.1. MongoDB Replica-Set

---

A MongoDB Replica-Set may be used to protect **Management Servers** deployments. The data is typically protected via replication between 3 servers, a primary and 2 secondary, see this [link](#) for more information.

There are some basic changes required in the `/etc/mongod.conf` file to

- 1.) Install MongoDB on different servers following the stand-alone instance instructions shown above. Make sure the servers are able to reach one another within the same well-connected data center for optimal latency and throughput to support the timely replication of data amongst the instances.

✿ With any MongoDB replica-set deployment it is important to ensure the configured service port, default tcp/27017, is accessible between all servers. The following instructions describe how to enable the MongoDB service to listen on an IP address reachable by the other instances and participate in the replica set. This change may also require modifications to firewalls on the network or running on the servers themselves. The following example assumes three servers with the “hostname (IP address)” set to nvme01 (172.10.100.201), nvme02 (172.10.100.202), and nvme03 (172.10.100.203).

! The MongoDB configuration file is formatted using YAML syntax and is very sensitive to mistakes. Pay close attention to the use of delimiters and spacing between options and indentation of nested configuration elements. Any format errors will lead to an inability to successfully start services.

On each node as root, edit the MongoDB configuration file:

```
vi /etc/mongod.conf
```

Modify the `bindIp:` field nested under `net:` by adding the desired endpoint IP after the loopback IP using a comma (no spaces) to separate the two addresses (nvme01 example):

```
net:
  port: 27017
  bindIp: 127.0.0.1,172.10.100.201
```

If you want the MongoDB to listen on all IP ports:

```
net:
  port: 27017
  bindIpAll: true
```

Uncomment the `replication:` field and set a `replSetName:` (indented with two spaces on the line directly beneath):

```
replication:
  replSetName: rs0
```

✿ Always define the same `replSetName` on all servers intended for the same replica-set.

On all three nodes, (re)start MongoDB (as root):

```
systemctl restart mongod
```

Verify services were started successfully on each of the three nodes (as root):

```
systemctl status -l mongod
```

Once verified, start the MongoDB shell and initiate replication by running the following commands on only one of the nodes (assuming nvme01):

```
# mongo
> rs.initiate()
rs0:PRIMARY> rs.add("172.10.100.202:27017")
rs0:PRIMARY> rs.add("172.10.100.203:27017")
```

While still on the MongoDB shell, validate the cluster formed successfully:

```
rs0:PRIMARY> rs.status()
rs0:PRIMARY> rs.config()
```

The status command will return an "ok": 1 message at the end of its output if all servers are up and running within the replication-set.

## 9.2.2. Update Management Servers Configuration

---

On each redundant **Management**, edit:

```
/etc/opt/NVMesh/management.js.conf
```

Update `config.mongoConnection.hosts` to match the hosts set up in the [MongoDB Replica-Set](#) section. The reference for this setting can be found in the [Management Server Options](#) section.

Update `config.mongoConnection.options.replicaSetName` to match the `_id:`, in the previous section example listed as `rs0`.

Example:

```
config.mongoConnection = {
  hosts: 'nvme01:27017,nvme02:27017,nvme03:27017',
  options: {
    replicaSetName: 'rs0'
  }
}
```

Perform the same operation for `config.statisticsMongoConnection` and `config.nvmeshMetadataMongoConnection`.

\* This set name must be the same on all redundant *Management Servers* in the cluster.

## 9.2.3. Using Load Balancers with HA Management

---

The use of a web load balancer may be incorporated into an nvmesh-management deployment in order to provide a single virtual IP (VIP) from which users may access any of the WebUI and API endpoints in an HA deployment.

In order for a load balancer to confirm that an endpoint is “In-Service”, it is common for the service’s developer to include a health-check URL. nvmesh-management exposes this feature at:

```
https://{listener_addr}:4000/isAlive
```

The expected response is:

```
Of course I'm alive..
```

\* Excelero Customer Support can provide guidance for deployment with the nginx load balancer.

# 10. Monitoring

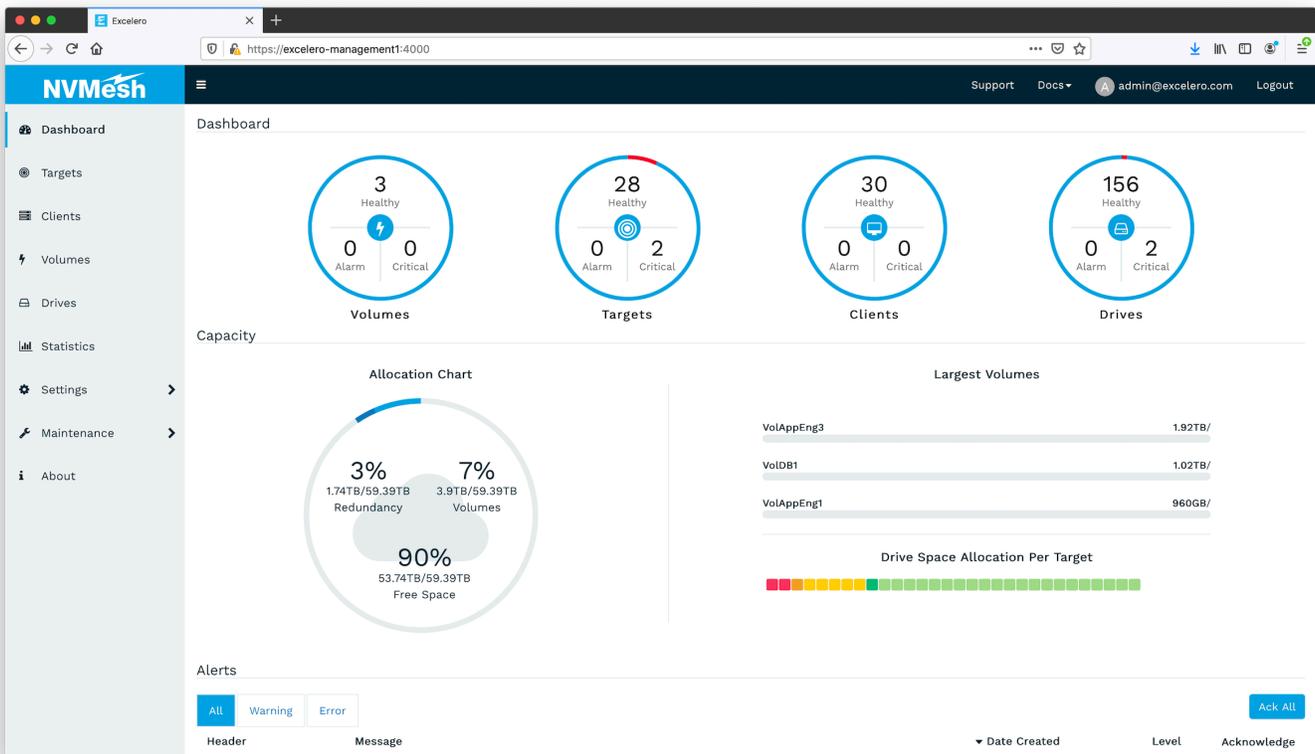
**NVMesh 2.5.2** provides various ways to monitor the system, both using the **Management's** GUI and extensive statistics data available in /proc.

Excelero also maintains a community supported telegraf plugin at [Telegraf and Grafana GitHub](#).

## 10.1. Dashboard

### Overview

The **NVMesh 2.5.2** dashboard provides a snapshot of overall cluster status. The dashboard is comprised of 3 areas, as follows.



### Overall Status Dashboard

The following gauges provide an overall status of the entire cluster:

- *Volumes*
- *Targets*
- *Clients*

- *Drives*

 Click on a gauge to open the relevant screen.

The 4 gauges are divided into health categories. Clicking on a health count for a specific leads to a list of entities for that element filtered by the health chosen.

## Capacity

The capacity sub-section provides the following graphical elements:

- *Allocation Chart*, which shows using a single gauge the effective space available, the amount of space used for redundancy and the remaining free raw space in the system.
- *Largest Volumes*, depicting the 4 largest volumes in the system.
- *Drive Space Allocation per Target*, depicts all targets and their current color-coded free space status .

## Alerts

Recent (non-acknowledged) alerts will show up here. A filter row is provided to aid in searching for alerts.

# 10.2. Volume State

A volume may be in various states. The current state is reflected in the *Management Servers*' GUI using two columns, **Action** and **Status**.

### Action

The **Action** column reports task or action related information on a volume. For the most part, the information provided is about the most recent action invoked that has not completed. The exception is **Rebuild Required** that calls for action from the administrator. The following table describes this information in more detail.

Volume Lifecycle	Action	Description
Allocation	<b>Initializing</b>	An administrator has defined a new volume. Not all relevant <b>Targets</b> have acknowledged that they have begun the process of creating the volume.
	<b>Extending</b>	An administrator has increased the size of the volume. Not all relevant <b>Targets</b> have acknowledged that they have begun the process of extending the volume. This is the parallel of the <b>Initializing</b> state for new volumes.

Steady State	<b>Rebuilding</b>	<p>The volume should be available for IO operations from <b>Clients</b>. The volume has some segments of data that may not be fully up-to-date with recent writes into the volume and these are currently in the process of being synchronized or rebuilt. There are enough <i>Drives</i> available for the entire volume data at the requested redundancy level. Once all synchronization is complete, the volume should transition to an <b>Online</b> state.</p> <p>A progress bar roughly tracks the rebuild. The progress should be considered an estimation.</p>
	<b>Booting</b>	<p>The volume is still not functioning normally and <b>Clients</b> will not be able to perform IO operations to this volume</p> <p>There are reports from relevant <b>Targets</b> that indicate that some of them have not yet completed the boot stage for this volume after a restart of the <b>Target</b>.</p>
Spare Space Allocation	<b>Rebuild Required</b>	<p>The volume has some segments of data that were on <i>Drives</i> that have been evicted whether by an administrator or automatically and a replacement <i>Drive</i> has not been allocated by management yet. To begin the allocation of <i>Drives</i> for rebuild, see <a href="#">Drive Failure and Replacement</a></p>
	<b>Marked for Rebuild</b>	<p>The volume has some segments of data that were on <i>Drives</i> that have been evicted whether by an administrator or automatically and a replacement <i>Drive</i> has been allocated by management. Not all relevant <b>Targets</b> have acknowledged the replacement so far.</p>
Deletion	<b>Marked for Deletion</b>	<p>An administrator has deleted the volume. It should not be available for IO operations from <b>Clients</b>, as detaching all <b>Clients</b> is a pre-requisite to volume deletion. Not all relevant <b>Targets</b> have acknowledged the replacement so far.</p>
	<b>Deleting</b>	<p>An administrator has deleted the volume. It should not be available for IO operations from <b>Clients</b>, as detaching all <b>Clients</b> is a pre-requisite to volume deletion.</p> <p>All <b>Targets</b> have acknowledged that they have begun the process of deleting the volume, but not all have completed it yet.</p>

## Status

The **Status** column reports the most recent status of a volume based upon reports from **TOMAs**. To see more detailed information on an individual volume and the status of the components that it is comprised of, click on the volume name to open up a diagram of the volume. Each element has its own **Status** field shown in a modal box opened when hovering over it with the mouse.

The following table describes the meaning of the volume statuses.

Status	Description
<b>Online</b>	The volume is functioning normally and should be available for IO operations from <b>Clients</b> . It has no known availability issues per the last report from <b>Targets</b> that have <b>Drives</b> on which data for this volume is stored.
<b>Offline</b>	The volume is not functioning normally and <b>Clients</b> will not be able to perform IO operations to this volume. It has known availability issues per the last report from <b>Targets</b> that have <b>Drives</b> on which data for this volume is stored.
<b>Degraded</b>	The volume should be available for IO operations from <b>Clients</b> . The volume does not have enough active <b>Drives</b> for the entire volume data at the requested redundancy level, but the active <b>Drives</b> do have the ability to provide all volume data that is up-to-date and therefore IO is operational.
<b>Unavailable</b>	There are missing reports from relevant <b>Targets</b> that make it impossible to determine the current state of the volume.

## Health

The volume's health is conveyed by the color of the volume, as described in the following table. In general, the health is a simpler summary of its state and action based on severity.

Color	Health	Description
<b>Green</b>	<b>Healthy</b>	This indicates a healthy volume naturally.
<b>Yellow</b>	<b>Alarm</b>	A volume's health is set to alarm when it is marked as requiring a rebuild, is rebuilding already or is in a degraded mode.
<b>Red</b>	<b>Critical</b>	A volume's health is set to critical when the status is offline or unavailable.

## Per-Client Volume State

When a volume is attached to a **Client**, there is also a state as seen by the **Client** that can differ from one to another and from that reported by **Management**, for instance depending upon the **Client**'s networking status.

**Clients** send status reports on their attachments to **Management Servers**. This is reflected in the **Clients** section of the GUI in the **Volume Attachments** column. For each **Client**, there is a list of the volumes to which it is attached. A green background color indicates that IO is enabled or functional, while a red one indicates that IO is disabled. When IO is disabled, follow the instructions at [Checking Volume Status](#) to get more information on the specific **Client**'s status for the specific volume.

If the volume is accessible via NVMe-oF from a specific **Client**, its status for the volume will include an icon depicting a plug.

## 10.3. Client State

---

The current **Client** state is reflected in the **Clients** table.

The **Volume Attachments** column lists the **Volumes** attached to the specific **Client**. The background color is used to reflect state.

- A **green** background indicates a successful attachment with IO enabled for the volume on this **Client** specifically.
- A **yellow** background indicates a successful and IO enabled attachment for a volume that is attached in a “hidden” form used only for rebuild functionality.
- A **red** background indicates that the attachment itself failed or that IO is currently disabled.

After configuration changes, for instance via *Configuration Profiles*, it is necessary to restart the **Client** to apply the changes. When the system recognizes this situation, a yellow warning triangle will appear to the left of the *Configuration Profile* name in the **Config Profile** column.

The **Health** column provides information on the livelihood of the **Client**. Hovering over the icon in the column will provide additional info.

- A checkmark within a **green** circle indicates normal functioning.
- An exclamation mark within a **yellow** triangle indicates that the **Client** is probably running, but reporting an error condition, such as volume with IO disabled or that it has not been communicating with the **Management** for more than 2 minutes, but less than 5.
- An exclamation mark within a **red** circle indicates an error state. Either the **Client** has been shutdown or it has not communicated with the **Management** for over 5 minutes.

## 10.4. Target State

---

The current **Target** state is reflected in the **Targets** table.

The state of a specific **Target** can be viewed by clicking on the relevant **Target ID**, which is the **Target**'s identifier generated from its hostname.

The **Drives** and **NICs** columns enumerate the number of each of these elements. Note that when a NIC name is changed or a drive is swapped out, the removed or deprecated elements may need to be deleted manually if the system is unable to determine that they are not simply missing.

After configuration changes, for instance via *Configuration Profiles*, it is necessary to restart the **Target** to apply the changes. When the system recognizes this situation, a yellow warning triangle will appear to the left of the *Configuration Profile* name in the **Config Profile** column.

The **Health** column provides information on the livelihood of the **Target**. Hovering over the icon in the column will provide additional info.

- A checkmark within a **green** circle indicates normal functioning.
- An exclamation mark within a **yellow** triangle indicates that the **Target** is probably running, but reporting an error condition, such as a missing *NIC* or *Drive* or the TOMA component’s status may be unavailable as it has not reported for a while.
- An exclamation mark within a **red** circle indicates an error state. Either a **Target** component has been shutdown or failed, for instance the TOMA component’s main process may have terminated, or it has not communicated with the **Management** for over 5 minutes.

When viewing a specific **Target**, it’s inventory is presented, *NICs* and *Drives*. Missing elements will be marked as such. The *Drives* are grouped as follows:

<b>Parity Ready</b>	These drives are ready for usage by <b>NVMesh</b> for any type of volume.
<b>Concatenated, Striped and / or Mirrored</b>	These are ready for usage by <b>NVMesh</b> for the volume types in the header, i.e. non-protected and mirrored.
<b>Not formatted for NVMesh</b>	These are available for usage by <b>NVMesh</b> , but have not been formatted.
<b>Excluded</b>	These are not available for usage by <b>NVMesh</b> , as they are in use by another application or excluded by the user.

## 10.5. Statistics

**NVMesh 2.5.2** provides customers with elaborate cluster-wide and per-object performance and utilization statistics that helps with the monitoring and analysis of the storage environment performance.

By default, **NVMesh 2.5.2** is configured to collect statistics from the entire cluster. In the *General* sub-section of the *Settings* section of the GUI, there is a general toggle option for collecting the system-wide statistics. Lower down there is also a *Statistics* configuration area.

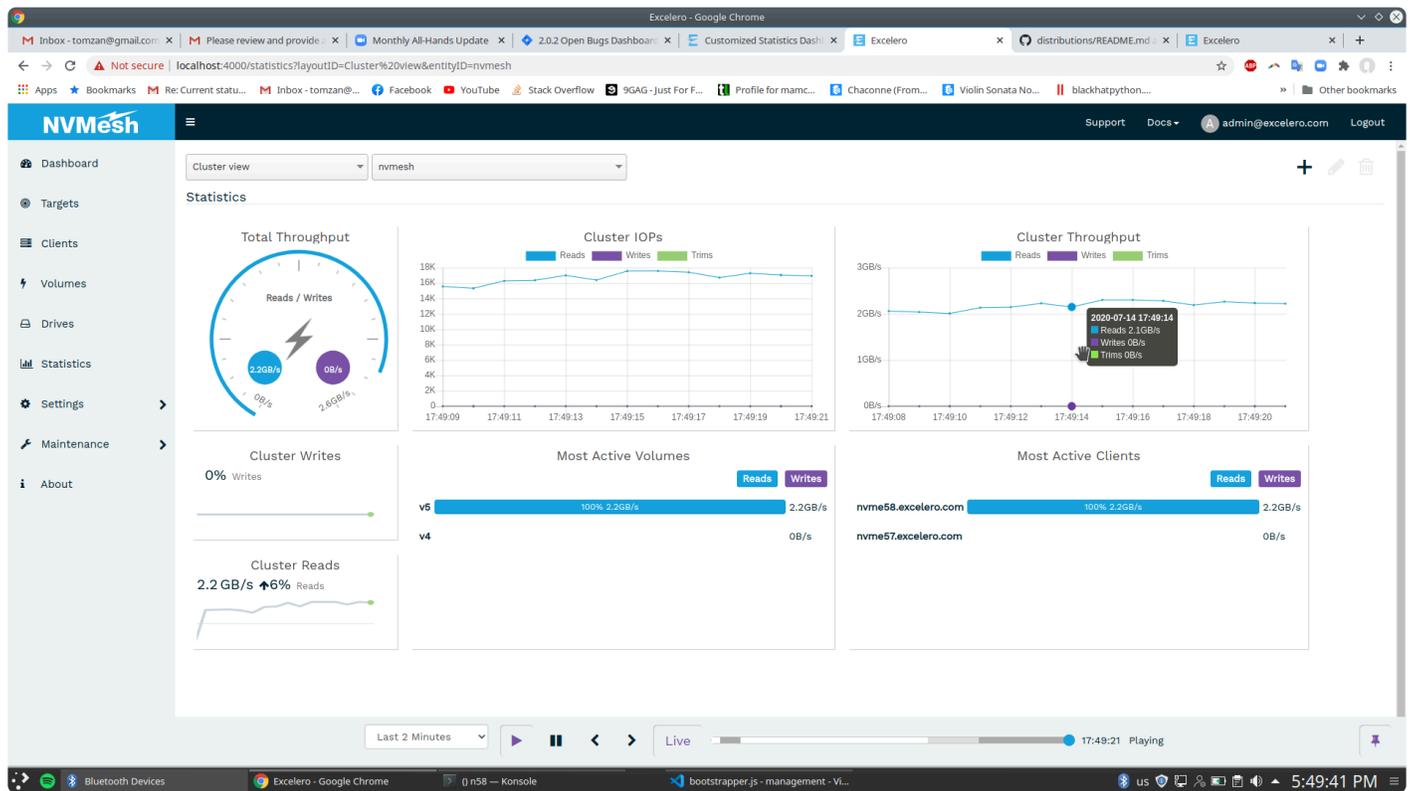
### 10.5.1. Customized Statistics Dashboard

The *Statistics* screen is a fully customized statistics dashboard. Various gadgets can be added to the dashboard, each with a selection of metrics to display. A default setup of gadgets is provided. The default view is the *Cluster view* providing cluster wide data and showing the activity hotspots. Three additional built-in-views provide a much more detailed view of specific **Targets**, *Drives* or *NICs*. When choosing one of these views, choose also the entity to depict.

The *Target view* shows the total network usage and total local drive usage. In addition, there is a graph per *Drive* and *NIC* on the target.

The *Drive view* provides both total throughput and IO/s gauges as well graphs for this data.

The **NIC view** provides a wealth of statistics, especially for Ethernet NICs, including throughput in general and specifically for RDMA and vendor specific data from hardware counters.



Additional installation-specific views are managed using the 3 icons at the top right of the statistics screen, which are from left-to-right, an ‘Add’, ‘Edit’ and ‘Delete’ button. All users have access to all views. The built-in views are not editable, but the rest can be edited as follows.

To add a gadget:

1. Unlock the statistics dashboard by clicking on the lock switch at the upper-right side.
2. Click the **Add Gadget** button.
3. Select the desired gadget from the drop-down list.

To remove a gadget:

1. Unlock the statistics dashboard by clicking on the lock switch at the upper-right side.
2. Click the settings icon at the upper-right side of the gadget.
3. Click the **Remove** button from the drop-down menu.

To move a gadget:

1. Unlock the statistics dashboard by clicking on the lock switch at the upper-right side.
2. Drag the gadget as needed. Surrounding gadgets will automatically adjust.

The **Bar Chart** and **Line Chart** gadgets can also be resized. To resize a gadget:

1. Unlock the statistics dashboard by clicking on the lock switch at the upper-right side.
2. Drag the shadowed boxes at the lower-left and lower-right corners of the gadget to resize the gadget as needed. Surrounding gadgets will automatically adjust.

Gadgets have multiple customization options helping drill down into graphs ranging from rather specific pieces of data to more complex aggregations.

## Line Charts

To define a line chart, choose one of the following entity types:

- Clients
- Targets
- Nodes
- Volumes
- Drives
- NICs

Then choose a specific entity of this type. Based on the entity type chosen, there may be an option to add a filter to further drill down. Choose all pieces of data to show, filling in the form following the order of red highlighted areas. Finally, at the bottom of the dialog, it is possible to set a custom name for the gadget.

## Dial Gauges

Defining a dial gauge begins with choosing an entity in the same manner done for the line chart and filling out the remaining fields.

Dial gauges also have an option for using auto-scale versus choosing a specific maximum.

## Spark Lines

Defining a spark line is similar to a line chart with a reduction in the number of pieces of data that can be shown.

## Bar Charts

Bar charts provide the ability to aggregate more data with a drill-down into entity specific information. First choose an entity type. However, for bar charts it is possible to choose all entities of the main entity type or the filtered entity type, for depiction in a single gadget.

In the bar chart , it is possible to limit the number of items shown and to choose them according to a specific sort order to allow drilling into hotspots.

## Health Counter Charts

For the health counter, choose one of the supported entities.

## Allocation Chart

This is a non-customizable widget showing the allocation of drive space across the system.

## 10.5.2. Statistics Gadgets

---

The following gadgets are available:

- Allocation chart
- Bar chart
- Dial gauge
- Health counter
- Line chart
- Spark line

## 10.5.3. Statistics Rollup

---

Statistics are collected by the **NVMesh 2.5.2 Clients** and **Targets** and sent to the **Management Servers**. For efficient use of resources, the collected statistics are rolled up based on the [rollup settings defined in the /etc/opt/NVMesh/management.js.conf configuration file](#). By default, there are 4 sampling intervals:

1. Statistics collected every 1 second are kept for 2 minutes, which means that the history is available at 1 second intervals up to 2 minutes back in time.
2. Statistics collected every 30 seconds are kept for 15 minutes, which means that the history is available at 30 seconds intervals up to 15 minutes back in time.
3. Statistics collected every 1 minute are kept for 1 hour, which means that the history is available at 1 minute intervals up to 1 hour back in time.
4. Statistics collected every 1 hour are kept for 1 day, which means that the history is available at 1 hour intervals up to 1 day back in time.

## 10.6. /proc Statistics

---

**NVMesh 2.5.2** provides customers with elaborate raw statistics in /proc that can be integrated with third-party monitoring software.

The information provided via this interface is prone to change or be enhanced without notice.

### 10.6.1. Client Volume Statistics

---

Filename: /proc/nvmeibc/volumes/<volname>/iostats

Provides a table with a column for each IO type (read, write & trim). Each row is a different metric. Before the table is the volume up\_time, which can be used to calculate rates.

•	Read	Write	Trim
<b>num_ops</b>	2	1	0
<b>size [bytes]</b>	8192	4096	0
<b>total_latency</b>	162.1	27.9	0.0
<b>total_execution</b>	164.7	4530.1	0.0
<b>latency^2</b>	0.0	0.0	0.0
<b>worst_latency</b>	83.3	27.9	0.0
<b>worst_execution</b>	84.1	4530.1	0.0
<b>worst_e2e</b>	1000	5000	0
<b>worst_e2e_enbl</b>	1000	5000	0

**num\_ops:** Amount of IOs kernel issued to the volume. Regardless of size of IOs and regardless of O operations of user space apps (user space app write of 1[mb] can be split by kernel to 8 writes of 128KB and volume will receive 8 IOs)

**size:** Total size of all the IOs in bytes. size/num\_ops will give you the average size of the IO.

The rest of the measurements are different types of latency. They are presented by the stages of IO execution:

- 0. An IO request is received by the **Client**'s kernel module
- 1. Optional: IO is delayed due to IO throttling, a the maximum number of outstanding IOs for this core has been reached
- 2. Optional: A few failed execution attempts of the IO, for instance. a disk was removed or a network connection lost during execution of the IO
- 3. Optional: Waiting because IO on the volume is disabled, for instance one of the disks of a non-mirrored volume is down or both legs of a RAID-1 are down
- 4. Successful execution
- 5. Return success to kernel

There are 4 interesting metrics, measuring ranges of those steps in units of microseconds:

- **io\_latency** – Measures step [4..5] – Reflects typical statistics under good path conditions, i.e. no disk or network issues
- **io\_execution** – Measures step [2..5] – Reflects the actually achieved statistics which may be worst due to disasters

- **io\_e2e** – End to end [**0..5**] – Comprises all steps. Note that although the number is in microseconds, the resolution is in milliseconds, so last 3 digits are 0
- **io\_e2e\_enbl** – End to end with IO enabled [**0..2, 4-5**] – This is all steps during which IO was enabled. If an IO got hung up because of ongoing external issue that won't be reflected here.

For each of the 3 metrics, there is interest in the total and in the worst.

**Total** is summation over all IOs to be used to calculate the average by dividing total by num\_ops or over a period of time by calculating deltas of these two parameters. **Worst** is the single IO which took the most time.

The table of stats is straightforward should be interpreted based on the above. Note that latency<sup>2</sup>\* is total or sum of all latencies squared, which can be used to calculate variance.

## 10.6.2. Client Drive Statistics

A volume is stored on one or more *Drives*. **Clients** attach to each *Drive* to implement accessing the volume. Statistics for the *Drives* used by the **Clients** are available in `/proc` at `/proc/nvmeibc/disks/<drive_name>/iostats.json`

This file has a JSON-formatted structure with the statistics of IO for this client with this *Drive*, for all volumes using it. The IO statistics are divided into objects by IO size.

Name	Value	Names	Value	Names	Value	Name	Value
uuid	Drive-Serial-Number (ex: P61024021537.1)						
uptime	Drive uptime in seconds						
stats	Object for various IO block sizes						
stats	Sub-Objects:	512B, 4K, 8K, 16K, 32K, 64K, 128K, >128K	Object for Statistics				
stats			Sub-Objects:	ops, size, latency	Object for Counters/ Values		
stats						read	Num
stats						write	Num
stats						trim	Num

overeager	Overeager Value						
-----------	-----------------	--	--	--	--	--	--

The JSON block has this structure:

```

{
  "uuid" : "P61024021537.1",
  "uptime" : "4398690.144",
  "stats" : {
    "<IOSIZE>" : {
      "ops" : {
        "read" : 0,
        "write" : 0,
        "trim" : 0
      },
      "size" : {
        "read" : 0,
        "write" : 0,
        "trim" : 0
      },
      "latency" : {
        "read" : 0.0,
        "write" : 0.0,
        "trim" : 0.0
      }
    },
    "overeager" : 0
  }
}

```

The values for <IOSIZE> are presented in the following table.

<IOSIZE>	IO Block Sizes
512B	1-512 bytes
4K	513 bytes to 4K
8K	>4K-8K
16K	>8K-16K
32K	>16K-32K
64K	>32K-64K
128K	>64K-128K
>128K	>128K and larger

Each <IOSIZE> has the following fields:

- **ops** Number of read/write/trim IOs the **Client** issued to the **Drive**.
- **size** Total size of read/write/trim IOs for this **Drive** in bytes.
- **latency** The total latency of read/write/trim IOs for this **Drive**, in microseconds.
  - latency/ops provides the average latency for this IO size.

Finally:

**overeager** – an internal (Excelero support and debug) term about **RDDA** channels where RDMA completion arrives to a **Client** before IO to the **Drive** had completed.

✿ The **Drive** statistics are cleared whenever the **Drive** is no longer needed to implement any volume. If it is later needed again, whether for the same or for a different volume, the counts will start from fresh.

## 10.7. /proc Monitoring

Monitoring information, largely Excelero debug information, can be found under various **/proc**. The following provides some basic information.

✿ As this information is intended primarily for support purposes, it is subject to change. It should not be considered as a steady system API.

### General Notes

- root permissions may be needed.
- Some items are write-only. Do not write into them without consulting [Excelero Technical Support](#).

Each sub-directory represents a module and has a version entry that reports version information in JSON format.

### /proc/nvmeib

This directory contains some general info common to **Clients** and **Targets**.

- alloc\_diag – used by developers to diagnose allocation issues.
- rdma/qp\_error – used by developers to simulate rdma QP errors.
- tracer/\* – a variety of endpoints used to monitor the nvmeshttracer service's behavior and configuration.

## /proc/nvmeiba

The nvmeiba module is shim layer between the operating system and the **Client** maintaining block device pointers enabling hot-upgrades to the underlying **Client**.

- status & status.json – provide status information on the block devices maintained, including path, number of opens & processes connected to them and outstanding IO counts (pending).
- users – a subset of the status focusing on opens & processes.

## /proc/nvmeibc

This directory is for **Clients**.

- cflags – presents the CFLAGS with which the module was compiled.
- cli/cli – used to implement communication between user space scripts and the kernel module. Writing to this file may cause unexpected results.
- dict\_sign – is a signature of the dictionary used for the tracer.
- disks – is a directory with a sub-directory for any disk the client is interacting with to implement any of the attached volumes.
  - <DRIVE-ID>/cmds – provides information of the Excelero commands sent to the drive since last connected to.
  - <DRIVE-ID>/counters – internal counters used to help diagnose incorrect behavior or performance issues.
  - <DRIVE-ID>/inj\_err – a mechanism for developers to inject artificial errors for debugging error paths.
  - <DRIVE-ID>/ioch – a subset of the status focusing on RDDA IO channels, communication paths to the drives.
  - <DRIVE-ID>/iostats & iostats.json – see [drive statistics](#).
  - <DRIVE-ID>/nrch – a subset of the status focusing on the non-RDDA IO channels, communication paths to the drives.
  - <DRIVE-ID>/rediscover – writing 1 into this file forces the **Client** to disconnect from the drive and “rediscover” it.
  - <DRIVE-ID>/status & status.json – multiple status entries regarding the *Drive*, including where it was last seen, **Target** software information, NICs used to communicate with it for the **Client** and **Target**, communication paths used and statistics on them, journal information for the *Drive* and locking statistics.
- dvlp – a facility for developers to interact with the **Client** using a CLI. Obsolete, will be removed.
- echo – for developers
- instctl – for user-space programs to control additional instances for multi-instance **Client** functionality.
- inst\_list.json – presents basic information for instance of the **Client** in use.
- jam/<DRIVE-ID> – in-depth information on journal area management per drive.
- mcs/mcs – used for communication with **Management**. Reading or writing from this entry may cause unexpected results.

- net/<NIC-ID>/iostats.json – provides statistics on NIC usage per IO operation type. This feeds into management statistics
- nics – information on the NICs used by the **Client**.
- rsrc\_info.json – information on the networking QPs used.
- status & status.json – general information about the **Client** that is more commonly accessed.
- volumes – a sub-directory per volume attached.
  - <VOL-NAME>/client\_processes – processes connected to and file opens on this volume's block device.
  - <VOL-NAME>/flow\_cntr.json – counters for various flows that recover volume data.
  - <VOL-NAME>/iostats & iostats.json – see [volume statistics](#).
  - <VOL-NAME>/io\_throttle – information on IO throttling
  - <VOL-NAME>/profiling & profiling.csv – used by developers for performance debugging.
  - <VOL-NAME>/recov\_stats – statistics on recovery processes.
  - <VOL-NAME>/status & status.json – multiple status entries regarding the volume and the status of its pieces. **This entry is widely used as a starting point to decipher why IO is disabled on a volume.**

## /proc/nvmeibs

This directory is for **Targets** and their embedded **TOMAs**.

- arnic\_prefer – used to control a mechanism for defining NIC preference per *Drive*.
- clients – outlines the **Clients** currently connected to this **Target**.
- disk\_info – a high level overview of the *Drives* and NICs as seen by the **Target** and used for reporting to **Management**.
- disks/<DRIVE-ID>/diag – basic info and statistics on *Drive* access.
- disks.csv – an alternative high level overview of the *Drives* as seen by the **Target**.
- dvlp – a facility for developers to interact with the **Target** using a CLI. Obsolete, will be removed.
- locks.<DRIVE-ID> – used by the **TOMA** to interact with a drive's locks. Reading or writing from this entry may cause unexpected results.
- log<DRIVE-NUM> – can be used to read the NVMe log of the drive enumerated as drive with this number. Use the smart proc entries to correlate with an NVMe serial id.
- mcs/mcs – used for communication with **Management**. Reading or writing from this entry may cause unexpected results.
- nic\_gids/<GID>.csv – description of NIC GIDs used as seen by the **Target**.
- nics.csv – an alternative high level overview of the NICs as seen by the **Target**.
- nvmeof\_disks – the NVMe-over-Fabrics drives that the **Target** can access. This can also be used to inform the system to use them, typically via an **NVMesh 2.5.2** user-mode script.
- partitions.csv – drive partition information.
- rsrc\_info.json – information on the drive resources used by **Client**.
- serjios.csv & serjio/<DRIVE-ID>/\* – serjio manages erasure-coded volume journal areas. These entries are used to monitor, debug and optimize its behavior.
- smart<DRIVE-NUM> – provides SMART output per drive enhanced with basic drive info such as PCI slot and serial ID. **Useful for locating a drive through the PCI address together with dmidecode**

**for hardware maintenance.**

- toma\_clients & toma\_servers – used to communicate between the **TOMA** and **Target**. Reading or writing from this entry may cause unexpected results.
- toma\_status – provides **TOMA** status. For most items, the most pertinent information is that at the **TOMA** leader.
  - all – a concatenation of all the other status entries.
  - bdev – the status of the block devices (volumes) as seen by this specific **TOMA**.
  - cfg – the configuration of volumes as received from **Management**.
  - disk – status of drives managed by all **TOMAs**.
  - dseg – status of all volume segments managed by all **TOMAs**.
  - leader – the RAFT leader, which makes cluster-wide volume status decisions.
  - local\_disk – status of the local drives, per **TOMA**, including partitions, which may indicate why a drive has been auto-evicted by **Management**.
  - mem – memory structures used, which can be useful for memory leak detection.
  - raft – status of the RAFT protocol used to choose a leader among the **TOMAs** in the zone. This will also show who the leader is if there is one. **The status on the leader itself can be used for diagnosis of inter-TOMA network communication issues.**
  - rdma – presents the status of communication channels to other **TOMAs** in the zone.
  - recover – status of volume recovery processes initiated or managed by this **TOMA**. The initiator is normally the **Target** with the up-to-date data.
  - topo – the system topology or volume statuses.

## 10.8. About Screen

---

The *About* screen provides information on various system attributes:



## Excelero NVMesh Management 2.2.0-18.el7\_8

© Copyright 2015-2020 Excelero, Inc. All rights reserved.

This document contains the confidential and proprietary information of Excelero, Inc. Do not reproduce or distribute without the prior written consent of Excelero.

[Read EULA](#)

### System Information

Management Version	2.2.0-18.el7_8
Cluster ID	My Cluster
EULA Signature	slash
Date of EULA Signature	02/01/2021 at 19:59:35
NodeJS Version	12.18.3
MongoDB Version	4.2
Replicated	false

[Download Status](#)

- **Management Version** – The *NVMesh 2.5.2 Management* version.
- **EULA Acceptance** – Who electronically accepted the End User License Agreement and when.

- **NodeJS Version** – The NodeJS version.
- **MongoDB Version** – The MongoDB version.
- **Replicated** – *true* if replication is enabled for high availability.
- **Download Status** button – Downloads a small JSON format file containing current cluster status.

## 10.9. Node Health Check (NHC)

---

The [Node Health Check](#) utility can be used to monitor local node activity.

Install the `nvmesh-nhc` package from the **NVMesh 2.5.2** repository. After installation, the NHC scripts for **NVMesh 2.5.2** will be in `/etc/nhc/scripts`. These checks should then be added manually to `/etc/nhc/nhc.conf`.

Issuing `nhc` will run the checks located in the `/etc/nhc/nhc.conf` file and can be used to validate status at any point in addition to the configured periodic NHC checks.

The current list of validation is as follows:

- `check_nvmesh_volumes_attached` <list of volume names>
- `check_nvmesh_volumes_status`
- `check_nvmesh_rdda_settings`
- `check_nvmesh_mongodb`
- `check_nvmesh_nvmeshmgr_service`

# 11. Logging

---

There are 3 mechanisms for logging within **NVMesh 2.5.2**. This section provides an overview of each and the means for controlling the granularity and severity of information sent to each. If there are issues in the system, [Excelero Technical Support](#) may request to increase the amount of logging generated. If the amount of logging is excessive, these means can help reduce the logging output.

The 3 mechanisms are:

- [System logs](#)
- [Management logs](#)
- [Binary tracing](#)

## 11.1. System Logs

---

**NVMesh 2.5.2** logs warnings and event to the standard system logs. This is done by all **NVMesh 2.5.2** components.

**NVMesh 2.5.2** kernel module logs will be available via the `dmesg` command line utility.

Kernel modules logs as well as logs from **Management**, the **TOMA** and other internal inter-component communication mechanisms such as **nvmeshcm** and **nvmeshagent** are sent to the general system log. For Redhat / CentOS, these are typically stored in `/var/log/messages*`. For Ubuntu, these are typically stored in `/var/log/syslog`. For both operating system flavors, the log messages should be available also via `journalctl`. System administrators may change the typical syslog definitions and manage them differently. Consult with a local system administrator if they are not available.

**NOTE:** On some RHEL-compatible distributions, the `rsyslog` package may be optional. If `/var/log/messages` is not present, this package may not be installed.

Following is a comprehensive list in alphabetical order of units related to **NVMesh 2.5.2** system logs. To view a specific unit's logs via `journalctl`, use `journalctl -u <UNIT>`, for instance `journalctl -u nvmeshmgr` to view the **Management's** system logs.

- `nvmeshagent` – the unit for the `nvmeshagent` service, which is the component that performs remote tasks on behalf of the management at **Client** and **Target** servers. Its main task is statistics collection.
- `nvmeshclient` – the unit for the kernel modules that provide the **Client** functionality.
- `nvmeshcm` – the unit for the `nvmeshcm` service, which is the component that communicates between the **Management** and the **Clients** and **Targets**. It runs on the same node as the **Clients**.
- `nvmeshmgr` – the unit for **Management Servers**.
- `nvmeshstats` – the unit for a sub-component of **Management** that deals with statistics collection.
- `nvmeshtarget` – the unit for the kernel modules that provide the **Target** functionality.
- `nvmesh toma` – the unit for the **Target's TOMA** component.

- `nvmeshtrace` – the unit for the [binary tracing](#) component.

See [software components](#) for more information on the components themselves.

By default, components will issue logs with levels of Info, Warning and Error.

For **Management**, the level of logging can be altered from `/etc/opt/NVMesh/management.js.conf` by configuring `logginglevel`, see [Management Log Level Options](#).

For the **nvmeshcm**, use the variable `MCS_LOGGING_LEVEL` in `/etc/opt/NVMesh/nvmesh.conf` respectively. Set it to one of the following `DEBUG`, `INFO`, `WARNING` or `ERROR`. Alternatively, set it to `VERBOSE` and then use `MCS_LOGGING_VERBOSE_TYPES` to control the specific verbose functionalities to debug at a verbose level. Consult with [Excelero Technical Support](#) in this case.

For the **nvmeshagent**, use the variable `AGENT_LOGGING_LEVEL` in `/etc/opt/NVMesh/nvmesh.conf`. Set it to `DEBUG` to increase the logging level.

The **Targets** and **Clients** are mainly implemented using kernel modules. Starting with **NVMesh 2.0**, most of their logs are maintained by the [binary tracing](#) mechanism. Any logs of level `INFO`, `WARNING` or `ERROR` will be forwarded also to the system log. Prior to **NVMesh 2.0**, logs had been sent directly to the system log and it was possible to control whether `TRACE` level logs were sent to the syslog using the standard kernel mechanisms or by setting a number of 2 or higher in `/sys/module/{nvmesh_module}/parameters/debug_level` where `nvmesh_module` can be one of `nvmeibc`, `nvmeibs` or `nvmeib_common`. This mechanism is being obsoleted and will be removed after all messages have been migrated to binary tracing.

## 11.2. Management Logs

---

**Management Servers** maintain logs of cluster-wide operations, not only management operations. These are not logs of the management software layer itself. They are accessible via the **Management's** GUI from the **Logs** subsection of the **Maintenance** section.

These logs are divided into `INFO`, `WARNING` and `ERROR` levels. Warning and error logs also generate alerts, see [Alerts](#) for more information on the contents of such logs.

A non-comprehensive list of the topics of Info logs are:

- Users logging in and out.
- Volume lifecycle: creation, extension, going offline, into a degraded state or back online, undergoing rebuild and deletion.
- **Target** lifecycle: detection, going offline or online, health issues and removal.
- **Drive** lifecycle: detection, formatting and implicitly being put into the drive pool, being removed from it, going offline or online, health issues, eviction and removal.
- NIC lifecycle: detection and going offline or online.
- **Client** lifecycle: detection, volume attachments and detachments and removal.

- *Volume Provisioning Group* lifecycle: reservation of *Drive* space for a *Volume Provisioning Group*.

## 11.3. Binary Tracing

---

Excelero introduced a new logging mechanism, starting from **NVMesh 2.0**, called “binary tracing”. This enables a fast lightweight highly controllable logging mechanism to enable tracing customer-side issues with ease. Out of the box, some binary traces are kept only in memory while others are also stored in persistent storage asynchronously. A drive sync is done every 2 seconds by default and is controlled by the `trace_dump_timeout` module parameter for the kernel module `nvmeib_common`. Data is stored in the directory `/var/log/NVMesh/trace_daemon`. If this directory is on a low-endurance drive, such as a typical boot drive, it may be recommended to use an alternative one.

Logs sent to binary tracing of levels `INFO`, `WARNING` or `ERROR` are also sent to standard system logs.

All binary traces are kept in memory and occasionally dumped to persistent storage. By default, more granular logs of lower severity than info-level logs, often called “trace”-level logs are also stored using this mechanism. The following module params control general tracing levels:

- `/sys/module/nvmeib_common/parameters/tracer_debug_level` – for the kernel modules common to **Clients** and **Targets**
- `/sys/module/nvmeibc/parameters/tracer_debug_level` – for the **Client** kernel modules
- `/sys/module/nvmeibs/parameters/tracer_debug_level` – for the **Target** kernel modules
- `/sys/module/nvmeibc/parameters/topology_debug_level` – for traces related to the volume’s topology topic in the **Client**
- `/sys/module/nvmeibc/parameters/goodpath_debug_level` – for traces related to implementing the **Client’s** good path IO
- `/sys/module/nvmeibc/parameters/goodpath_locks_debug_level` – for traces related to implementing the **Client’s** good path locking operations

The values used for these module parameters are as follow:

- “1” – only errors will be kept by the binary tracer and forwarded to the syslog.
- “2” – only errors and warnings will be kept by the binary tracer and forwarded to the syslog.
- “3” – errors, warning and info level traces will be kept by the binary tracer and forwarded to the syslog.
- “4” – errors, warning and info level traces will be kept by the binary tracer and forwarded to the syslog. Trace-level messages will be kept by the binary tracer only.
- “5” – errors, warning and info level traces will be kept by the binary tracer and forwarded to the syslog. Trace-level and debug messages will be kept by the binary tracer only.
- “6” – errors, warning and info level traces will be kept by the binary tracer and forwarded to the syslog. Trace-level, debug and fine messages will be kept by the binary tracer only.

If changing these parameters, it is advised not to go below debug level 3, as setting debug levels 1 or 2 will make it very hard to troubleshoot issues.

To change the memory and drive footprint for the **Client** and **Target** kernel modules, use the following options settable in `/etc/opt/NVMesh/nvmesh.conf`:

- `TRACE_BUFS_PER_LOG` – specifies the number of 4K buffers saved to a single trace file. The default value of 4096 translates into 16 MB files. It is not advised to modify this value.
- `TRACE_MAX_LOGS` – specifies the number of files of history to keep. The default value of 64 translates to 1 GB in total (when `TRACE_BUFS_PER_LOG="4096"`). The minimal value can't be lower than the number of CPUs on the server (lower values will be ignored). To adjust the size of history on the disk, it is best to adjust this value only.

To change the drive footprint for the **TOMA** component, use the following options settable in `/etc/opt/NVMesh/nvmesh.conf`:

- `TOMA_TRACE_LOG_SIZE` – specifies the size of a single TOMA trace file. The default value of 48 translates into 48 MB files. This parameter can be between 1-200.
- `TOMA_NUM_OF_TRACE_LOGS` – specifies the number of files of history to keep. The default value of 40 translates to about 1.8 GB in total (when `TOMA_TRACE_LOG_SIZE="48"`). This parameter can be between 1-100.

For additional instructions on how to view binary traces and how to control their footprint, contact [Excelero Technical Support](#).

# 12. Alerts

**NVMesh 2.5.2** alerts administrators on important topics. The alerts appear in the lower half of the dashboard screen in a table format. There are 2 levels of alerts, *Warning* and *Error*. Use the **NVMesh 2.5.2** tabular GUI to review the alerts. Administrators can acknowledge alerts to remove them from the main dashboard, either individually or using the *Ack All* button. Some alerts are acknowledged automatically by the system once the condition has been resolved. All alerts including those acknowledged can be seen in the **Logs** subsection of the **Maintenance** section.

## Error Alerts

Header	Message	Comments
Target node failure	Target: <b>&lt;node name&gt;</b> is up, but requires attention	This occurs when the <b>Target</b> has a malfunctioning NIC or <i>Drive</i>
Target node failure	Target: <b>&lt;node name&gt;</b> is up, but the NVMesh Target is down	This is a result of stopping the nvmeshtarget service without stopping the nvmeshclient service
Target node failure	Target: <b>&lt;node name&gt;</b> is up, but the NVMesh TOMA is down	This is a result of stopping the nvmeshtoma service only or the nvmeibt_toma executable being terminated
Target node failure	Target: <b>&lt;node name&gt;</b> is down	This is a result of no report being received from the target node for a period exceeding the timeout period defined for management, which is configurable
Drive failure	Drive: <b>&lt;drive id&gt;</b> reported status: <b>&lt;drive status&gt;</b> and health: <b>&lt;drive health&gt;</b>	Drive status is <i>Format Error</i> or <i>Offline</i> . Drive health is <i>OK</i> , <i>Warning</i> or <i>Critical</i>
NIC failure	NIC: <b>&lt;NIC descriptor&gt;</b> reported error	The NIC appears to be in an error state on the host
NIC failure	NIC: <b>&lt;NIC descriptor&gt;</b> is missing	The NIC can no longer be found on the host
Format incomplete	Drive <b>&lt;drive id&gt;</b> format failed	
Critical drive endurance	Drive SN <b>&lt;drive id&gt;</b> endurance is below 1%	
Volume status is	Volume <b>&lt;volume name&gt;</b> status changed to offline	One or more of the drives comprising the volume is unavailable or down so that the volume is currently effectively offline.

offline		
---------	--	--

## Warning Alerts

Header	Message	Comments
Drive automatically evicted	The drive: <b>&lt;drive id&gt;</b> was automatically evicted for the following reason: <b>&lt;reason&gt;</b>	<p>Possible reasons for auto-eviction are:</p> <ul style="list-style-type: none"> <li>• <i>Drive reports an invalid size or changed its size</i>, in regard to the size that the <b>Target</b> reported for this <i>Drive</i></li> <li>• <i>Serial is 'UNKNOWN' or null</i>, in regard to the Serial ID read from a <i>Drive</i> by a <b>Target</b></li> <li>• <i>Invalid Partition Table</i>, reported for <i>Drives</i> that have a non-NVMesh generated partition table</li> <li>• <i>Missing Partition Table</i>, reported for <i>Drives</i> that post NVMesh format did not have a partition table</li> <li>• <i>Partition size was changed</i>, reported for <i>Drives</i> for which the partition size differs from the one expected, which was put on the <i>Drive</i> previously</li> <li>• <i>Overlapping segments found</i>, which means that the metadata for this <i>Drive</i> is in an illegal state. It is recommended to contact <a href="#">Excelero Technical Support</a></li> <li>• <i>Segment out of bound</i>, which means that the metadata for this <i>Drive</i> is in an illegal state. It is recommended to contact <a href="#">Excelero Technical Support</a></li> <li>• <i>System partition found on drive</i>, implying that an external entity added a partition to the <i>Drive's</i> partition table</li> <li>• <i>Orphan NVMesh data partition found</i>, implying that the <i>Drive's</i> partition table refers to a piece of a <i>Volume</i> that does not exist in the management database</li> <li>• <i>Drive formatted with volume segments</i>, which means that a <i>Drive</i> appears to be newly formatted, although it had already been in use for <i>Volumes</i></li> <li>• <i>Data partition was found after format</i>, implying that a <i>Drive's</i> partition table refers to a piece of a <i>Volume</i> immediately after being formatted</li> <li>• <i>Drive is missing suitable metadata partitions</i>, reported for <i>Drives</i> for which one of the NVMesh internal partitions was deleted</li> <li>• <i>Unknown NVMesh partition found on drive</i>, implying that the <i>Drive's</i> partition table has an NVMesh created partition without an NVMesh partition type</li> <li>• <i>Drive uuid mismatch found</i>, which means that the <i>Drive's</i> NVMesh identifier is different than that in the management database</li> <li>• <i>Drive was imported from another NVMesh environment</i></li> </ul>

		<ul style="list-style-type: none"> <li>• <i>Drive reported an invalid uuid</i>, meaning that the <i>Drive's</i> NVMesh identifier is invalid</li> <li>• <i>Error while processing drive</i>, which is the default error, but also implies that the <b>Target</b> had an error when reading the <i>Drive's</i> metadata</li> </ul>
Low drive endurance	Drive SN <b>&lt;drive id&gt;</b> endurance is below <b>&lt;N&gt;</b> %	<b>N</b> is one of 5, 10, 20 or 50
Duplicate NIC ids	Received multiple NICs with the same ID from node: <b>&lt;node name&gt;</b>	
Duplicate drive ids	Received multiple drives with the same ID from node: <b>&lt;node name&gt;</b>	
Mirror violation detected	Drive relocation caused mirror violation for volume: <b>&lt;volume name&gt;</b>	This means that the mirrored copies of data on a mirrored volume, R-1 or R-10, are now co-located in a target node or violating a protection domain requirement. For erasure coded volumes, this implies that separation requirements have been violated, which could be too many copies in the same target node or violating a protection domain separation requirement.
MTU is higher than 4200	MTU is higher than recommended for RoCE/Infiniband on NIC <b>&lt;NIC descriptor&gt;</b> in node <b>&lt;node name&gt;</b>	4200 is the recommended value in general for RDMA. This may not be applicable to other environments. Contact <a href="#">Excelero Technical Support</a> for instructions for suppressing these alerts if the standard in the network exceeds this
Rebuild Required	Volume <b>&lt;volume name&gt;</b> requires rebuild	This means that the volume had data on an evicted drive and as such should be rebuilt. See <a href="#">Drive Failure &amp; Replacement</a>
Volume status is degraded	Volume <b>&lt;volume name&gt;</b> status changed to degraded	The volume has data on a drive that became unavailable, so the volume is available, but data availability is degraded

# 13. NVMesh Best Practices

---

There are various choices, settings and practices in hardware and software in the **NVMesh 2.5.2** infrastructure that affect performance, reliability and failover. While this section provides various suggestions for different environments, it is not a substitution for [Excelero Technical Support](#) nor is every suggestion applicable to all situations. When in doubt, contact Excelero Technical Support, [support@excelero.com](mailto:support@excelero.com).

## 13.1. Performance Best Practices

---

The following section describes various performance optimization best practices. It is highly recommended to follow these best practices to achieve the lowest IO latencies.

### 13.1.1. Homogeneous OFED Configuration

---

In RDDA configurations, it is important to keep the OFED version consistent across the cluster for best performance. RDDA will not be employed across heterogeneous OFED versions.

### 13.1.2. CPU Interrupt Affinity and IRQ Balancing

---

IRQ balancing and NUMA affinity for Mellanox NICs is an advanced topic. For more complete information, please see the Mellanox document [Performance Tuning for Mellanox Adapters](#).

The instructions below are general guidelines that apply to both **NVMesh 2.5.2 Clients** and **Targets** using Mellanox RDMA adapters, Ethernet or Infiniband.

Mellanox provides the `mlnx_affinity` and `mlnx_tune` tools to solve for tuning adapter IO interrupts for optimal balance and NUMA-local affinity.

#### OS-Native IRQ Balancer

The OS-native IRQ Balancer service alone can often be sufficient.

You can verify the service is running with the `systemctl` command as follows:

```
systemctl status irqbalance
● irqbalance.service - irqbalance daemon
   Loaded: loaded (/usr/lib/systemd/system/irqbalance.service; enabled; vendor preset: enabled)
```

```

Active: active (running) since Fri 2018-01-19 08:58:28 PST; 1 weeks 2 days ago
Main PID: 1493 (irqbalance)
CGroup: /system.slice/irqbalance.service
        └─1493 /usr/sbin/irqbalance --foreground

Jan 19 08:58:28 uslab-11.uslab.excelero.com systemd[1]: Started irqbalance daemo
n.
Jan 19 08:58:28 uslab-11.uslab.excelero.com systemd[1]: Starting irqbalance daemo
n...
```

If `irqbalance` appears to remain an issue despite the service being active or performance is low, especially if one or more CPU threads are being pegged to 100% during IO, then it is recommended to test using one of the previously mentioned Mellanox tools.

Further investigation is best accomplished by reading out the `/proc/interrupts` file and verifying the adapter interrupts are well distributed amongst the CPU threads. Additionally, ensure the list of CPU threads that are considered NUMA-local to the adapter, see the `mlnx_tune` output as shown below, are the only ones being used. Use of non-NUMA-local threads will require the use of the CPU interconnects (e.g. UPI or Infinity Fabric) and may impose additional latency thus reducing peak performance. Both of the Mellanox tools are designed to assign NUMA-local threads for adapter IO. Determining which tool works best may require testing.

## Mellanox Affinity (optional)

To enable `mlnx_affinity` by default, edit the following file:

```
/etc/infiniband/openib.conf
```

Add or modify the line:

```
RUN_AFFINITY_TUNER=yes
```

Also, disable the `irqbalance` service permanently. To stop and then disable:

```
systemctl stop irqbalance
systemctl disable irqbalance
```

## Mellanox Tune (optional)

In addition to being useful in setting interrupt affinity and balance, this tool may also be used to report on the status of the adapters in terms of PCIe connection details as well as system memory and CPU performance settings. Here is an example of its output when run without options for setting any tuning:

## Mellanox Technologies - System Report

## Operation System Status

UNKNOWN

3.10.0-693.5.2.el7.x86\_64

## CPU Status

Intel Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz Broadwell

Warning: Frequency 2600.0MHz

## Memory Status

Total: 124.56 GB

Free: 116.67 GB

## Hugepages Status

On NUMA 1:

Transparent enabled: always

Transparent defrag: always

## Hyper Threading Status

ACTIVE

## IRQ Balancer Status

NOT PRESENT

## Firewall Status

NOT PRESENT

## IP table Status

NOT PRESENT

## IPv6 table Status

NOT PRESENT

## Driver Status

OK: MLNX\_OFED\_LINUX-4.2-1.0.0.0 (OFED-4.2-1.0.0)

## ConnectX-5 Device Status on PCI 85:00.0

FW version 16.21.2010

OK: PCI Width x16

>>> PCI capabilities might not be fully utilized with Broadwell CPU. Make sure I/O non-posted prefetch is disabled in BIOS.

OK: PCI Speed 8GT/s

PCI Max Payload Size 256

PCI Max Read Request 4096

Local CPUs list [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 42, 43,

```

44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]

ens3 (Port 1) Status
Link Type eth
OK: Link status Up
Speed 100GbE
MTU 4200
OK: TX nocache copy 'off'

2018-01-29 08:07:23,638 INFO System info file: /tmp/mlnx_tune_180129_080720.log

```

In the example above, note the `System info file:` mentioned at the end of the output. This file contains additional details and recommendations for further tuning the system, if applicable.

Aside from reporting the configuration data as shown above, you may also apply the recommended `HIGH_THROUGHPUT` profile as follows:

```
mlnx_tune -p HIGH_THROUGHPUT
```

The output will be similar to the example, but the resulting output file shows that it has balanced and set processor affinity for the adapters in a long listing at the end of the file. Here is an example of that section of the file:

```

Ring Info:
  rps_mask: 000000,000000000
  xps_wanted_mask: N/A
  number: 55
  xps_mask: 000200,000000000
  rps_wanted_mask: N/A
  active: True
  IRQs:
    IRQ Info:
      wanted_mask: N/A
      smp_affinity_mask: 000200,000000000
      number: 472
      affinity_hint_mask: 000200,000000000

```

## 13.1.3. Tuned

From the Red Hat RHEL 7 [Power Management Guide](#):

Tuned is a daemon that uses `udev` to monitor connected devices and statically and dynamically tunes

system settings according to a selected profile.

Tuned is distributed on Red Hat and CentOS Linux distributions with pre-defined performance profiles. It is part of the `tuned` package. For the best **NVMesh 2.5.2** performance results, choose the `latency-performance` profile. This setting should be applied to both **Clients** and **Targets**. To enable this profile on Red Hat Enterprise Linux and CentOS distributions, use this command, as root: `tuned-adm profile latency-performance`.

To verify or display the current active tuned profile use `tuned-adm active`.

Which should display: `Current active profile: latency-performance`

Changes made with `tuned-adm` should survive system reboots.

## 13.1.4. NUMA Architectures

---

### Overview

When setting up **NVMesh 2.5.2** on servers with non-uniform memory access (NUMA) based CPU architectures, it is important to consider the configuration of NICs and NVMe drives versus the NUMA. Since each CPU core has a specific, fixed throughput ceiling in its interconnect path to any one of the NUMA memory regions (nodes), and each CPU has a fixed number of PCIe lanes to NICs and NVMe drives, an IO issued from a CPU core to a specific NIC or NVMe drive may be impacted by this NUMA interconnect throughput.

As a result, IO sent from an application running across the CPU cores may suffer from inconsistent performance because some IOs will be issued against a local NUMA node, while other IOs will be issued against a remote NUMA node, with a much slower throughput.

Therefore, it is best to balance the memory accesses according to the specification of the CPU cores and NUMA nodes, the interconnect performance, and the performance specifications of the NICs and NVMe drives.

actual specifications of the CPUs,

### Review NUMA Topology

It is possible to review the NUMA nodes and CPU cores in `lscpu`. The [Portable Hardware Locality](#) project provides the `lstopo` utility which lists the NUMA nodes, storage and network devices in a single output.

To install the utility:

```
yum install hwloc
```

To run the utility:

```
lstopo-no-graphics --ignore PU --merge --no-caches
```

## Possible Configuration Changes

- Physically move NICs in between PCI slots. This depends on the motherboard's NUMA configurations.
- Physically move NVMe drives in between PCI slots. This depends on the motherboard's NUMA configurations.
- Set applications to run on specific cores, using `taskset` or `numactl` commands. to optimize the connectivity between the application IOs and the destination NICs and NVMe drives.

! For AMD EPYC chips, PCI to PCI operations, often called peer-to-peer DMA, do not guarantee order with a single NUMA. Therefore, to use *RDDA* on such chips, it is imperative to separate the NICs and the NVMe drives to separate NUMA nodes.

## arnic\_prefer

**NVMesh 2.5.2** has a mechanism for assigning preferred NICs for *Drives* for manual optimization. This means that IO operations for the *Drive* will be sent via the specific NIC as much as this possible. For instance, in network failure scenarios, other NICs could be used. This mechanism coined **arnic\_prefer** is controlled via `/proc/nvmeibs/arnic_prefer`. Contact [Excelero Technical Support](#) for detailed guidance.

## 13.2. Multi-Path Configuration

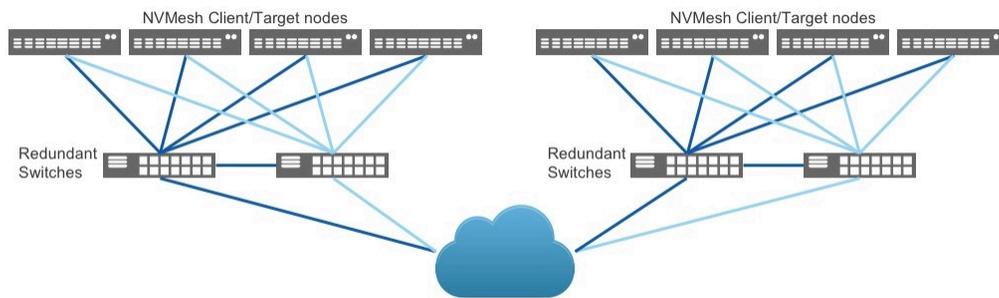
---

Multi-path network configuration may be desired for bandwidth aggregation, failover, or both. It is common practice to leverage standard dual-port networking components to provide networking redundancy. Dual discrete single-port components are also typically deployed. Due to performance considerations, multiple networking ports may also be deployed within a single device.

This section lays out recommendations for configurations that should enable **NVMesh 2.5.2** to achieve high performance and high availability in multi-port environments.

At the heart of current best practices are the following:

- Each node for which redundancy or high performance is required should have links to distinct switches.
- Switches should be interconnected.
- Prefer routed links.
- No spanning tree.
- Intelligent broadcast and multicast control.



Redundant Network Paths

## 13.2.1. Ethernet & RoCE

### General Considerations

For Ethernet networks with IP as a layer 3, it is recommended to implement multiple LANs of limited size and to perform full routing, i.e. ensure reachability from any end-point to any end-point.

\* It is important to note that RoCE does not work properly with link aggregation protocols such as LACP – you must implement host based routing to utilize redundant Ethernet links.

From the vantage point of a single end-point, it is best to hook up the machine to different switches to avoid a single point of failure. Cross-switch LANs are in contradiction to avoiding spanning trees and broadcasts. Therefore, it is best to use different LANs per port, avoiding host-side bonding mechanisms in general.

**NVMesh 2.5.2** leverages RDMA to achieve its extraordinary capabilities. **NVMesh 2.5.2** has been certified on devices employing RDMA-over-converged Ethernet (a.k.a. RoCE). There are two flavors of RoCE. RoCE v1 is non-routable and not supported. RoCE v2 is certified and as a routable protocol fits well with the best practices described above.

Configuring layer 3 routing on current state-of-the-art switches is trivial and comes at a negligible performance hit if any.

On the network end-points, **NVMesh 2.5.2** can leverage the ability to generally reach from any port to any port to generate multiple paths between **Clients** and **Targets**. This requires configuring source-based routing at the end-points so that packets will exit via the port associated with the end-point address chosen for communication for both sending and receiving packets. The mechanism for doing this differs from operating system to operating system and is also dependent on whether the NetworkManager service is active.

Another aspect of RDMA is the usage of the RDMA CM (connection manager facility). Some NIC software stacks separate the settings for the RoCE protocol used by general RoCE messaging and the RDMA CM messaging.

## Mellanox ConnectX Ethernet Adapter Considerations

Mellanox ConnectX series NICs are certified for **NVMesh 2.5.2**. For Ethernet, only NICs supporting RoCE v2 are formally supported. **NVMesh 2.5.2** upon startup will ensure that the `RDMA_CM` mode is set to match.

If applying pre-installation tests to ensure RDMA and RDMA\_CM functionality, setting this mode using the `ma_roce_mode` utility provided in OFED is recommended.

## 13.2.2. RDMA Atomics Limitations

---

Mellanox ConnectX-4, ConnectX-5 and ConnectX-6 NICs normally perform RDMA atomic operations within the NIC. Therefore, RDMA atomic operations received on different NICs or performed locally on a CPU are not atomic to each other. **NVMesh 2.5.2** relies on RDMA atomic operations for efficiency. Even while enduring failover, it is imperative that all clients can reach the same NIC (either port) on a target machine.

At least two dual-port target server NICs are required for path failover and NIC failure protection in multi-path environments. **Clients** should be configured from a routing standpoint to be able to reach both ports of each NIC.

## 13.3. Hardware Related Items

---

Choices in NVMe Drives, NIC cards and PCIe slot or channel assignment can have a significant impact on system performance and system limitations. This subsection contains information specific to choices in hardware and hardware combinations.

### 13.3.1. NVMe Drives

---

#### NVMe Devices

**NVMesh 2.5.2** should be operable with any drive that adheres to the NVMe 1.0 or higher specification.

The current list of certified hardware can be found in the [Interoperability Matrix](#).

Please notify Excelero if you plan to use or test a drive that is not listed there.

#### NVMe IO Queues

The ultra-low latency and extreme performance characteristics of **NVMesh 2.5.2** are best achieved when **Clients** can make direct use of the IO queues on the remote NVMe drives. While the NVMe specification permits as many as 65,536 (64K) IO queues, current drives typically have implemented only 8, 16, 32 or 128 queues. **Clients** work more efficiently when they can reserve IO queues on remote NVMe devices. If they

cannot get dedicated IO queues from a given NVMe drive it will still work, but not as efficiently.

This situation may occur when there are many **Clients** utilizing one physical NVMe device. For example, if there is a single volume that is striped across two **Targets**, each with a single 1TB NVMe drive, using all available space, and it is attached to one **Client**, it is free to utilize all the available IO queues on both NVMe devices. If however, the volume is attached to 16 **Clients**, they would be in competition with each other for the IO queue resources. If instead there were NVMe drives that had only 32 IO queues, each **Client** could only utilize 2 queues.

Alternatively, if there was a single **Target** with a single 3.2TB NVMe drive and you wanted to split this up into scratch space for many physical or virtual **Clients**, you might split the disk into 64, 50GB volumes. If each of these volumes was assigned to a unique **Client** and your NVMe drive had only 32 IO queues, performance on all the **Clients** would be hindered due to a lack of sufficient queues. With NVMe disks with 128 queues, each **Client** could get at least 2 queues.

## 13.3.2. PCIe Considerations

### PCIe

Use a following like the one in the following example to obtain the relevant information for PCIe device after identifying its bus enumeration. 85:00.01 in the example.

```
[root@nvme241 20:23:16 ~]$ sudo lspci -vvv -s 85:00.01 | grep -e LnkSta: -e LnkCap:
                LnkCap:          Port #0, Speed 8GT/s, Width x16, ASPM not supported, Exit Latency L0s unlimited, L1 unlimited
                LnkSta:          Speed 8GT/s, Width x16, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
```

- LnkCap refers to the device capabilities.
- LnkSta refers to its current status.

### PCIe Generation

Physical servers used for **NVMesh 2.5.2**, especially **Targets**, should utilize PCIe Generation 3 or higher. If performance issues are encountered, it is recommended to verify whether the NIC is an appropriate slot matching its PCI generation. Also, it is recommended to verify that NICs and NVMe drives are running in practice using at the required generation, as the hardware elements often negotiate this. This can be verified using `lspci`, as described above and comparing the Speed data.

### PCIe Slot Width

It is recommended to verify that NICs and NVMe drives are running in practice with the PCIe slot width they have been inserted to, as the hardware elements often negotiate this. This can be verified using `lspci`, as

described above and comparing the Width data.

## NVMe drives

Currently available NVMe drives utilize PCIe x4 or x8 interfaces. Drives that utilize x16 or more lanes are reportedly under development. Placing an NVMe drive in a slot with fewer lanes than the drive's rating will typically reduce its performance, primarily in terms of read bandwidth.

## NICs

Slot allocation for NICs need to be taken into consideration in system design. Typically, a NIC should be inserted into a slot that at least matches its rating to enable it to achieve its maximum bandwidth. Most high-speed NICs utilize an x8 interface or an x16 interface.

Care must be taken in system design and NIC selection regarding aggregate link speed of a card's network ports. For example, a dual-port ConnectX-4 or ConnectX-5 NIC 100GbE card can link both ports at 100Gb/s, or an apparent 200Gb/s in aggregate. The card has an x16 PCIe interface and hence is limited to 100Gb/s. Thus, when designing a system for maximum performance capability and multi-path/multi-port aggregation is desired for higher total bandwidth, utilizing multiple separate single-port 100Gb/s NICs will allow the NICs themselves to be unrestricted by PCIe bus speeds.

## PCIe Multiplexers

Some hosts employ PCIe multiplexers, usually to increase the number of NVMe drives that can be inserted into the system with each drive having sufficient lanes of PCIe-3 or PCIe-4 bandwidth. This will enable accessing each individual drive at its maximum bandwidth, but the system will not be able to utilize all drives at their maximum simultaneously. System designs should be aware of these elements and limitations when estimating total potential performance.

 **NVMesh 2.5.2**, thanks to patented *RDDA*, provides a unique benefit for overall bandwidth in new system designs that connect both NICs and NVMe drives to such multiplexers, when used with drives with NVMe CMB functionality.

## QPI/UPI

Typically, Intel QPI/UPI bandwidth and AMD Infinity Fabric, i.e. inter-CPU connectivity, is lower than PCIe bandwidth for each CPU. Therefore, to optimize a system, it is recommended to avoid data crossing between CPUs as much as possible. **NVMesh 2.5.2** has tuning facilities to help in this functionality. Contact [Excelero Technical Support](#) for more details.

## 13.3.3. Network Interface Cards

---

The definitive list of Network Interface Cards that are certified for use with **NVMesh 2.5.2** can be found in the [Interoperability Matrix](#).

In general, for RDMA based connectivity, NICs from Mellanox are recommended.

For TCP/IP based connectivity, it is important to utilize ports with enough bandwidth to support the desired storage traffic load. 4 NVMe drives can usually completely saturate a 100 Gb/s link.

For any NIC, please make sure to consider aggregate link bandwidth of the network ports vs. the PCIe server connection as detailed in [PCIe Considerations](#).

## 13.4. Network Considerations

---

Overall system performance may be governed by network limitations. This section gives some guidelines from the **NVMesh 2.5.2** perspective.

### Multi-switch Topologies

In multi-switch topologies, the overall bandwidth of the edge nodes may surpass that of the core of the network. This is largely dependent upon the network topology chosen. Localizing volume access may be pertinent to maximize performance. [Target Classes](#) and [VPGs](#) are tools for administrating rack-local or similar access patterns.

Sometimes dual switch setups are used for high availability. Due to [RDMA atomic limitations](#), it is highly recommended to use dual-port NICs instead of multiple single-port NICs. In a dual switch setup, if they are interconnected, **NVMesh 2.5.2** will indiscriminately send traffic cross-switch and intra-switch. Therefore, the interconnect between the switches may become a performance bottleneck if not adequately provisioned.

 In previous versions, it was imperative to use dual-port NICs to have RDMA atomic failover. This is no longer the case and **NVMesh 2.5.2** will fail over with redundant single-port NICs.

### Dedicated Storage Network

It is possible to separate **NVMesh 2.5.2** storage traffic from other traffic, even if the other traffic is RDMA. This can be done by selecting specific network elements for use with **NVMesh 2.5.2** via the appropriate [NIC configuration options](#).

## Infiniband

There are currently no specific Infiniband performance recommendations.

## Ethernet

### Global Pause

Global pause is the simplest way to implement lossless ethernet traffic useful for efficient RDMA. It is pertinent mainly with dedicated storage networks.

### Priority Flow Control

Priority flow control is a means for implementing lossless ethernet traffic useful for efficient RDMA. It enables mixing of storage traffic using RDMA and other traffic flows without compromising the non-storage traffic to the lossless networking limitations.

### ECN

ECNs, i.e. early congestion notifications, are a means for avoiding lossless network configuration while still providing relatively efficient RDMA, as packet drops are largely avoided. This is a good means for large traffic configurations in which lossless traffic engineering is not accepted.

Using a combination of PFC and ECN is also recommended, as the combination of both tends to work better at scale than each mechanism individually.

For lossy networks, i.e. those in which only ECNs are used as a congestion prevention mechanism or those without any congestion prevention implemented, it is recommended to employ Mellanox's Lossy RoCE.

## 13.5. Kernel Configuration

---

There are various kernel configurations that are recommended to achieve performance and stability from *NVMesh 2.5.2*.

### Serial Console

*NVMesh 2.5.2* uses the operating system's central log, see [System Logs](#).

For some events, *NVMesh 2.5.2* may generate messages at a high rate. If a serial console is used, the central log should be configured to suppress sufficiently or the serial console should be run at a reasonable speed, i.e. running at the default 9600 baud rate without suppression may be an issue. If the message rate is too fast for the serial console, soft and hard lockups may occur. The serial console is usually enabled

from the kernel boot command line using `console=ttyS0`. To increase the speed, use `console=ttyS0,115200n8`.

## Free Memory Space

The default operating system setting for free space left for the kernel may be too low. Increasing the value may be warranted, especially for **Clients** running workloads with high parallelism, multiple threads with high IO depth, especially if this is done in parallel on multiple volumes.

If encountering operating system slowness in such situations, it is recommended to increase the value for `/proc/sys/vm/min_free_kbytes`.

To make the change permanent, use `sysctl`. Insert a statement such as, `vm.min_free_kbytes=<VALUE>` in a conf file at `/usr/lib/sysctl.d/`.

## 13.6. MongoDB Best Practices

---

The **Management Servers** rely on the MongoDB database. Following are guidelines or recommendations for configuring the underlying database.

 When restarting MongoDB, for instance when making configuring changes, **Management Servers** should be stopped prior to MongoDB restart and then restarted after the restart has completed. This helps avoid any consistency issues.

### Authentication

For security purposes, it may be prudent to add an authentication layer to protect the data in the management database.

See [Enable Access Control for MongoDB](#) for instructions for configuration authentication to enable access control for MongoDB.

See [MongoDB Connection Options](#) for guidance on configuring authenticated access to MongoDB from the **Management Servers**, specifically `config.mongoConnection.auth.username`, `config.mongoConnection.auth.password` and `config.mongoConnection.auth.authenticationDatabase`.

### Memory Constraints

By default, MongoDB will use up to 50% of available RAM on the node. Typically, a few GBs of memory are sufficient for efficient **Management** behavior.

See [storage.wiredTiger.engineConfig.cacheSizeGB](#) for instructions on how to reduce the amount of memory taken by the database beyond standard operating system caches. Reducing it to 8GB is the nominal recommendation.

## High Availability

It is recommended to use MongoDB replica sets to ensure high availability of the data and of the database service.

See [Replica Set Configuration](#) for detailed instructions on how to setup MongoDB for high availability. See [MongoDB Connection Options](#) for guidance on configuring access to a highly available MongoDB instance with replicaSets, specifically `config.mongoConnection.hosts` and `config.mongoConnection.options.replicaSetName`.

# 13.7. Best Practices for AMD EPYC Processors

---

This section provides a comprehensive list of recommendations for AMD EPYC processors. All recommendations are based on a storage focused point-of-view. For servers used in a converged manner including, but not limited to, file system serving, there may be additional tweaking or optimization required for best performance.

## BIOS Settings

The following fields may differ slightly between server vendors and server BIOS versions.

### Processor Settings

#### Logical Processor (Hyper-threading) = Enabled

Enabling hyper-threading enables **NVMesh 2.5.2** to run more tasks in parallel. This has been found empirically to improve performance especially for smaller block sizes and for erasure coding write flows.

#### L3 cache as NUMA Domain = Enabled

This setting reduces expensive thread migrations across NUMA zones and reduces cache trashing. Disabling this has been found empirically to reduce performance.

#### NPS = 1 (dual-socket systems) or NPS = 4 (single-socket systems)

This setting **with L3 cache as NUMA domain enabled** helps ensure that multiple memory channels are active in parallel enabling higher bandwidth.

### System Profile Settings

#### CPU Power Management = Maximum Performance

#### C States = Disabled

#### X2APIC = Enabled

#### Memory Frequency = Maximum Performance

## DRAM Refresh Delay = Performance Memory Interleave = Auto

It may also be possible to set these in software, using tools like `tuned-adm`, e.g. using `tuned-adm profile latency-performance`.

## IOMMU

**NVMesh 2.5.2** does not support environments with AMD IOMMUs enabled. There is typically a BIOS setting for this.

Alternatively, set `amd_iommu=off` as a linux kernel boot option. Use `cat /proc/cmdline` to view the current boot options.

## Interrupt management

As AMD EPYC CPUs have multiple NUMAs per socket, interrupt management, which typically keeps interrupts within the NUMA node associated with the physical device and its PCI bus location, is too restrictive. Therefore, the system's `irqbalance` rarely performs optimally and it is recommended to disable this and use the following guidelines instead.

### NVMe drives

For the NVMe drives, it is recommended to use one of the following two new options for interrupt management, through `nvmesh.conf`.

1. Set the number of queues in use from each NVMe drive to be that of a physical processor, or as high as possible if there are not enough queues. Then, set `ASSIGN_NVME_IRQS="persocket"` in `nvmesh.conf`.
2. Set the number of queues in use from each NVMe drive to be that of both physical processors in a dual node system, or as high as possible if there are not enough queues. Then, set `ASSIGN_NVME_IRQS="fullspread"` in `nvmesh.conf`.

Changes in the above can be implemented without restarting services, by manually running `nvmesh_set_irq_affinity`.

### NICs

For NVIDIA Mellanox NICs, it is recommended to verify best practices with their support team.

Excelero has found empirically, that setting the interrupts to all cores of the same physical processor to which the NIC is PCI-connected is optimal.

This can be set using the `set_irq_affinity_cpulist.sh` utility. Note that this utility does not persist the settings. Use `rc.local` and similar mechanisms to ensure it is run on startup.

For instance, if NIC `mlx5_1` is connected to physical CPU 1 and this processor comprises logical cores 16-31 and 48-63, use:

```
set_irq_affinity_cpulist.sh 16-32,48-63 mlx5_1
```

For other NICs and TCP, follow the NIC vendor recommendations.

## Other NVIDIA NIC settings

It has been found empirically that for **NVMesh 2.5.2**, setting `PCI_WR_ORDERING` to relaxed for all queue pairs has major performance benefits without infringing on correctness. If other software solutions use the same NICs, it is imperative to ensure that the relaxed ordering is acceptable also for those software solutions.

Setting this is done using standard NVIDIA tools, such as:

```
mstconfig -d mlx5_0 set PCI_WR_ORDERING=1 or
mlxconfig -d mlx5_0 set PCI_WR_ORDERING=1
```

## NVMe drives

RDDA does not work well with AMD processors. Therefore, it is recommended to use many NVMe queues as possible on each drive, up to 1 per core. There is no need to exceed this. Use the target's module parameter for this, e.g. for a system with 64 cores add such lines a modprobe configuration file.

```
options nvmeibs min_local_nvmeqs=32
options nvmeibs max_local_nvmeqs=32
```

By default, RDDA is disabled. The *Configuration Profile* for the **Target** can be used to change this setting. It is an advanced setting for **Targets**.

To override the *Configuration Profile* setting, set `MLX5_RDDA_ENABLED="No"` in `/etc/nvmesh/nvmesh.conf`.

## Others

For heavy workloads, especially with many drives and many **Targets**, it is recommended to enable per-cpu polling. This is true for Intel processors as well.

This can be setting through module parameters, for instance by adding such lines to a modprobe configuration file.

```
options nvmeibs use_pcpu_cq=Y that sets this functionality for Targets.
options nvmeibc use_pcpu_cq=Y that applies to Clients.
```

[Excelero Technical Support](#) can also provide additional help in tuning for specific workloads.

# 14. Maintenance

---

The following section describes information on Excelero's support, describes available troubleshooting tools, and provides detailed instructions for various hardware and software related maintenance procedures that may be required during the ongoing operations of **NVMesh 2.5.2**.

## 14.1. Support

---

Excelero offers multiple methods to obtain support for **NVMesh 2.5.2**. Additionally, it includes support scripts, described in the next section, in the `nvmesh-utils` package for setup validation and log collection.

### Email

The preferred contact method to obtain support is to send an email to [support@excelero.com](mailto:support@excelero.com)

Doing so will generate a ticket or case number that can be used to track your issue and is the best way to ensure all needed information is in one location and easily accessed by all Excelero support personnel. Excelero has staff that monitor incoming support requests 24/7.

### Phone

[Excelero Technical Support](#) are available at this number:

**+1 888 6NVMESH (+1 888 668-6374)**

Select option '1' for support. If a support representative is not immediately available, leave a voicemail and a support representative will return your call within the terms of your support agreement.

## 14.2. Troubleshooting

---

### 14.2.1. `nvmesh_health_check`

---

`nvmesh_health_check` is a script used to validate the status of services and important OS attributes & settings.

The purpose of this script is to quickly validate the OS configuration and health of **NVMesh 2.5.2** services after initial installation. It may also be used later to confirm no changes to the system have occurred in these areas since their rollout.

To run `nvmesh_health_check` use `sudo` or run as root:

```
sudo /usr/bin/nvmesh_health_check
```

Following is an example of the output on a healthy system:

```
Mellanox Adapter Model / Firmware Info:
0000:03:00.0 mlx5_0 (MT4119 - MCX516A-GCAT) CX516A - ConnectX-5 QSFP28 fw 16.25.1
020 port 1 (ACTIVE) ==> eno1 (Up)
0000:03:00.1 mlx5_1 (MT4119 - MCX516A-GCAT) CX516A - ConnectX-5 QSFP28 fw 16.25.1
020 port 1 (ACTIVE) ==> eno2 (Up)
0000:85:00.0 mlx5_2 (MT4119 - MCX516A-GCAT) CX516A - ConnectX-5 QSFP28 fw 16.25.1
020 port 1 (ACTIVE) ==> ens3f0 (Up)
0000:85:00.1 mlx5_3 (MT4119 - MCX516A-GCAT) CX516A - ConnectX-5 QSFP28 fw 16.25.1
020 port 1 (ACTIVE) ==> ens3f1 (Up)

Installed OFED Release:
MLNX_OFED_LINUX-4.6-1.0.1.1:

Firewalld Service Status:
  Active: inactive (dead)

Check SELinux Status:
SELinux is Disabled as required.

Check Active tuned Profile:
The latency-performance tuned profile is correctly set.

IRQ Balancer Service Status:
  Active: active (running) since Mon 2020-02-17 15:55:08 IST; 11h ago

Display nvmesh-core package:
nvmesh-core-2.0.0-156.x86_64

NVMesh-target status:
Detected Kernel: 3.10.0-957.27.2.el7.x86_64
Installed Ofed: MLNX_OFED_LINUX-4.6-1.0.1.1
NVMesh-target Version: 2.0.0-156

All modules up

Managed NVMe Drives by Serial Number:
S3HCNX0K500688, S3HCNX0K700672
ManagementCM process running
Management Agent process running
Toma process running
nvmeshtarget status [ OK ]
```

```

NVMesh-client status:
Detected Kernel: 3.10.0-957.27.2.el7.x86_64
Installed Ofed: MLNX_OFED_LINUX-4.6-1.0.1.1
NVMesh-client Version: 2.0.0-156

All modules up

Attached Volumes:
EcVol1_8_2, EcVol2_6_2
Management Agent process running
ManagementCM process running
nvmeshclient status [ OK ]

Check nvmesh-management version and service status:
nvmesh-management-2.0.0-13.x86_64
NVMesh-management(5030) is listening on port 4000
NVMesh-management(5030) is listening on port 4001
NVMesh-management is up!
nvmeshmgr status[ OK ]

TOMA Leader:
nvme243.excelero.com

```

Following is an example of the output of the script under less than favorable conditions. For some of the items, there is an explanation on why they are not optimal and possible remedies.

```

Mellanox Adapter Model / Firmware Info:
0000:03:00.0 mlx5_0 (MT4119 - MCX516A-GCAT) CX516A - ConnectX-5 QSFP28 fw 16.25.1
020 port 1 (ACTIVE) ==> eno1 (Up)
0000:03:00.1 mlx5_1 (MT4119 - MCX516A-GCAT) CX516A - ConnectX-5 QSFP28 fw 16.25.1
020 port 1 (ACTIVE) ==> eno2 (Up)
0000:85:00.0 mlx5_2 (MT4119 - MCX516A-GCAT) CX516A - ConnectX-5 QSFP28 fw 16.25.1
020 port 1 (ACTIVE) ==> ens3f0 (Up)
0000:85:00.1 mlx5_3 (MT4119 - MCX516A-GCAT) CX516A - ConnectX-5 QSFP28 fw 16.25.1
020 port 1 (DOWN ) ==> ens3f1 (Down)

Installed OFED Release:
MLNX_OFED_LINUX-4.6-1.0.1.1:

Firewalld Service Status:
Active: inactive (dead)

```

Check SELinux Status:

SELinux is Disabled as required.

Check Active tuned Profile:

Please set the tuned-adm profile to latency-performance.

```
(run as root) # tuned-adm profile latency-performance
```

IRQ Balancer Service Status:

```
Active: active (running) since Mon 2020-02-17 15:55:08 IST; 11h ago
```

Display nvmesh-core package:

```
nvmesh-core-2.0.0-156.x86_64
```

NVMesh-target status:

```
Detected Kernel: 3.10.0-957.27.2.el7.x86_64
```

```
Installed Ofed: MLNX_OFED_LINUX-4.6-1.0.1.1
```

```
NVMesh-target Version: 2.0.0-156
```

```
All modules up
```

Managed NVMe Drives by Serial Number:

```
S3HCNX0K500688, S3HCNX0K700672
```

```
ManagementCM process not running
```

```
Management Agent process running
```

```
Toma process not running
```

WARNING: Unoptimized ConnectX-5 configuration. Performance will be degraded.

In order to fix, issue the command below and power cycle:

```
# mlxconfig -d 0000:03:00.0 -b /etc/opt/NVMesh/Excelero_mlxconfig.db set ONE_QP_P  
ER_RECOVERY=1
```

```
nvmeshtarget status [FAILED]
```

NVMesh-client status:

```
Detected Kernel: 3.10.0-957.27.2.el7.x86_64
```

```
Installed Ofed: MLNX_OFED_LINUX-4.6-1.0.1.1
```

```
NVMesh-client Version: 2.0.0-156
```

```
All modules up
```

Attached Volumes:

```
EcVol1_8_2, EcVol2_6_2
```

```

Management Agent process running
ManagementCM process running
nvmeshclient status [ OK ]

Check nvmesh-management version and service status:
nvmesh-management-2.0.0-13.x86_64
NVMesh-management isn't listening on the port: 4000
NVMesh-management is down!
nvmeshmgr status[FAILED]

TOMA Leader:
nvme243.excelero.com

```

## 14.2.2. nvmesh\_logs\_collector

/usr/bin/nvmesh\_logs\_collector

Create a log bundle for use by [Excelero Technical Support](#). This utility collects general system configuration information.

```

usage: python nvmesh_logs_collector.py [-h] [-c] [-f] [-i] [-a] [-g] [-d] [-D]
                                     [--all-collections] [-S JOURNALCTL_SINCE_DATE]
                                     [-n N] [-z]
                                     [--keywords [KEYWORDS [KEYWORDS ...]]] [-s]
                                     [--customer-name CUSTOMER_NAME]
                                     [--customer-eng-contact CUSTOMER_ENG_CONTACT]
                                     [--customer-eng-phone CUSTOMER_ENG_PHONE]
                                     [--customer-eng-email CUSTOMER_ENG_EMAIL]

```

NVMesh log collector script

optional arguments:

```

-h, --help                show this help message and exit
-c                        collect last crash file
-f                        collect full syslog files (don't filter)
-i                        ignore syslog files
-a                        collect syslog archived files (gz or bz2)
-g                        collect toma core file using gcore
-d                        do not dump database to disk
-D, --dump-blocked-tasks-state
                          dump uninterruptible (blocked) tasks state

```

```

--all-collections      include all collections while dumping the DB
-S JOURNALCTL_SINCE_DATE
                        collect from journalctl entries not older than the
                        specified time in seconds (default is 2h ago: -7200)
-n N                  choose number of lines to collect from journalctl
-z                    run the nvmesh_clnt_analyzer tool to collect extra
                        client logs
--keywords [KEYWORDS [KEYWORDS ...]]
                        additional keywords to collect from syslog in addition
                        to the default keywords
-s                    copy NVMesh source files (client and target)
--customer-name CUSTOMER_NAME
                        adds the customer name into the customer_info file
--customer-eng-contact CUSTOMER_ENG_CONTACT
                        adds the customer engineer contact into the
                        customer_info file
--customer-eng-phone CUSTOMER_ENG_PHONE
                        adds the customer engineer phone name into the
                        customer_info file
--customer-eng-email CUSTOMER_ENG_EMAIL
                        adds the customer engineer email into the
                        customer_info file

```

There is generally no need to pass any of the optional arguments unless specifically instructed to do so by [Excelero Technical Support](#).

## 14.3. Hardware Operations

---

The following section provides procedures for common hardware related maintenance operations.

### 14.3.1. Drive Failure & Replacement

---

**NVMesh 2.5.2** provides volumes with data protection to reduce the probability of data loss when *Drives* fail and to increase storage access availability.

With **NVMesh 2.5.2**, *Drive* failure detection and subsequent recovery is performed by **Targets**.

Upon *Drive* failure, the **Targets** will direct **Clients** attached to volumes affected by the failure to move to a degraded mode. In this mode, **Clients** will read and write from the remaining copy for MeshProtect 1 & 10 volumes. For MeshProtect 60 volumes, reads will be done by reading the other data blocks in the stripe and one or more parity blocks. Writes will only update the parity blocks in the degraded mode. For unprotected volumes, IO will become disabled. Multiple failures may lead to IO becoming disabled for protected

volumes.

Upon *Drive* failure, the **Target** associated with the *Drive* will send an alert to one of the **Management Servers** that will appear in the **Dashboard** section, in the [Alerts](#) section, with a *Drive failure* header. In the dashboard, the **Target**'s health will be in the **alarm** state, unless it is **critical** for some other reason. The same health state will be shown in the **Targets** section. Drilling into the **Target** will show the specific *Drive* that has failed using a white exclamation mark on a red background graphic. Hovering over this graphic will display a string describing the *Drive* failure.

If a failed *Drive* cannot be recovered, it can be replaced with an alternate *Drive* or with sufficient spare capacity on other *Drives*. The steps to replace a *Drive* are to evict the failed *Drive* and then rebuild the data to an alternative *Drive* area.

\* There is no need to replace a *Drive* with a specific replacement *Drive*. Space can be allocated from other *Drives* in the system that meet the provisioning criteria of the degraded volume, i.e., mirrored copies must be on different hosts and *Drives* used for erasure coded volumes should meet the volume's defined separation and location criteria.

## Evicting a *Drive*

In the **Targets** section, click on the **Target** with the *Drive* to be evicted. Then, click on the red **Evict** button on the *Drive*. A pop-up window will prompt for a password reflecting the sensitivity of this operation. Enter the password to proceed. For a failed *Drive*, the red exclamation mark icon, should be replaced with a yellow exclamation mark and hovering on it will display the *Disk Evicted* message. The **Evict** button is no longer accessible.

Evicted *Drives* cannot be deleted, like any other *Drive*, until all allocated space on them has been migrated to replacement *Drives* or has been disassociated from volumes by deleting the volumes.

The same functionality is also available from the **Drives** section.

## Volume Rebuild

Any protected volumes that have space allocated on an evicted *Drive* and no other issues, will have a status of *Rebuild Required*. In the **Volumes** section, such volumes can be located by entering "Rebuild" in the **Status** filter box. To rebuild or more volumes, use the volume table's multi-choose functionality and click on the **Rebuild** button in the top-left corner. A pop-up window will prompt for a password reflecting the sensitivity of this operation. Enter the password to proceed and the rebuild process will be invoked.

\* Note: If the volume had been defined using constraints, these will be applied by the system in choosing capacity for the rebuild process.

- If the volume was defined using a [Volume Provisioning Group](#), rebuild space will be allocated from it.
- If the volume was defined using some combination of [Target Classes](#) and [Drive Classes](#), rebuild space will be allocated from these classes, including space added to them after the original volume definition.
- If the volume was allocated from specifically chosen *Drives*, it may be necessary first to redefine the volume definition constraints.

Once the rebuild process is invoked, **NVMesh 2.5.2** will begin to copy data to the replacement capacity allocated.

New *Drives*, whether as a replacement *Drive* or in general, need to be formatted before they can be used. See [Format Drives](#).

## 14.3.2. NIC Failure and Replacement

---

**NVMesh 2.5.2** provides network high availability options, using multi-pathing and support for multiple NICs. Upon replacing a NIC, it may be necessary to take one or more of the following steps to ensure optimal behavior.

### General

- Ensure that the new NIC is defined as usable in the configuration for this node. The definition may be set in a *Configuration Profile* for this node, see [Configuration Profiles](#) or directly in `/etc/opt/NVMesh/nvmesh.conf`, see [Configuration Options](#).
  - Changing the configuration will require restarting the **NVMesh 2.5.2** services on the node.

### RoCE Specific

- If the NIC is in a **Target**, ensure that the malfunctioning NIC no longer appears in the GUI for this **Target** by deleting it. If the new NIC has the same IP address, this step should not be required.

### Infiniband Specific

- If the NIC is in a **Target**, ensure that the malfunctioning NIC no longer appears in the GUI for this **Target** by deleting it. Unless the NIC has the same GUID burnt into its firmware, this step will be required.

 If the replaced NIC is not deleted, **Clients** and other **Targets** will continue to attempt to connect to it potentially wasting network resources.

## 14.3.3. Moving a Drive

---

*Drives* can be freely moved within the same **Target**, or in between **Targets** of the same cluster. However, *Drives* cannot be imported into other **NVMesh 2.5.2** clusters. Such *Drives* will be considered foreign and will be automatically evicted by the system and an appropriate alert will be raised.

If a *Drive* is moved to the same **Target** as another *Drive* that holds data that is mirrored from it for one of the mirrored volume types, this will cause a mirror violation, which will be reported as an alert. This will not affect I/O, but will reduce availability. The same applies for erasure-coded volumes and aggregating *Drives* in a **Target** beyond the settings for the volume. Other protection domain separation criteria may also be violated in this way, but they are not alerted on.

\* It typically takes a number of seconds for drive removal and insertion operations to be completed. Therefore, it is recommended to wait at least 60 seconds between steps if reverting an accidental drive move, i.e. between removing a drive that had been recently inserted it into a different **Target** and then moving it back to the original one.

## 14.3.4. Drive Resize

---

Some *Drives* enable the administrator to modify their size. This operation if performed without the necessary preparations on the **NVMesh 2.5.2** side, may cause data loss. It is important to perform the resize using the following instructions, as it is presumed that resize operations will literally erase all data. If this is not the case, consult with [Excelero Technical Support](#).

1. Evict the *Drive* as if it was a failed drive, see [Drive Failure & Replacement](#).
2. Delete the *Drive*.
3. Resize the *Drive*, using the *Drive* vendor's instructions.
4. Remove and then reinsert the *Drive* physically to make **NVMesh 2.5.2** rediscover the drive.  
Alternatively, this can done via the operating system using the system's PCI manipulation commands. Be sure to correctly identify the *Drive* and as root perform `echo 1 > /sys/bus/pci/devices/<PCI_DEVICE>/remove` to remove the drive and then as root perform `echo 1 > /sys/bus/pci/rescan` to rediscover it or electronically reinsert it.
5. Format the drive for **NVMesh 2.5.2** use.

**Note:** if the *Drive* is not evicted, but is resized, it should be auto-evicted and an alert will be generated.

## 14.4. Software Operations

---

The following section provides procedures for common software related maintenance operations.

## 14.4.1. Start NVMesh

---

To start an **NVMesh 2.5.2** cluster, it is recommended to perform the following actions in the order as listed.

### Start All Management Services

Start the `nvmeshmgr` service on all **Management Servers**. This can be performed concurrently.

To start the `nvmeshmgr` service using the Linux CLI:

```
systemctl start nvmeshmgr
```

### Start All Target Services

Start the `nvmeshtarget` service on all **Targets**. This can be performed concurrently.

To start the `nvmeshtarget` service using the Linux CLI:

```
systemctl start nvmeshtarget
```

### Start All Client Services

Start the `nvmeshclient` service on all clients. The `nvmeshclient` service is automatically started on **Targets** when the `nvmeshtarget` service is started.

This can be performed concurrently.

To start the `nvmeshclient` service using the Linux CLI:

```
systemctl start nvmeshclient
```

## 14.4.2. Shutdown NVMesh

---

To properly shutdown an **NVMesh 2.5.2** cluster, it is recommended to perform the following actions in the order listed.

1. Stop the `nvmeshclient` service on all **Clients**. This can be performed concurrently using `systemctl stop nvmeshclient` from the Linux CLI.
  - a. If there are mounted filesystems using **NVMesh 2.5.2** block devices, unmount these filesystems first.
2. Stop the `nvmeshclient` service on all **Targets**. This can be performed concurrently using `systemctl stop nvmeshclient` from the Linux CLI. Stopping the `nvmeshclient` service will also stop the `nvmeshtarget` service that is dependent on it.
  - a. Alternatively, to stop all **Targets** from the management GUI:
    - i. Click **Maintenance**, then click **NVMesh Cluster**
    - ii. Click the red on/off button next to **Shutdown NVMesh cluster nodes**
3. Optionally, stop the management service on all **Management Servers**. This can be performed concurrently using `systemctl stop nvmeshmgr` from the Linux CLI.

## 14.4.3. Uninstall the NVMesh Software

---

Prior to the uninstallation of **NVMesh 2.5.2** software, stop all processes. For more information refer to [Shutdown NVMesh](#) including the optional phase of stopping the management services.

Once all services are stopped, the **NVMesh 2.5.2** packages can be uninstalled.

To remove all software use:

1. `yum remove nvmesh-utils nvmesh-core nvmesh-management` for RHEL/Centos distributions
2. `apt-get --purge remove nvmesh-utils nvmesh-core nvmesh-management` for Ubuntu distributions

## 14.4.4. Modify Management Server IP

---

When the IP address associated with a **Management** is modified, it will refuse to start and an error message like the following will be logged in `/var/log/messages` or `/var/log/syslog`, depending on operating system distribution:

```
Sep  5 14:35:44 nvmeshstorage nvmeshmgr[11107]: WARNING: Unable to verify managementId, sleeping for 10 seconds
```

To update the management server ID, create a file that indicates to the management that it should update its ID. A simple means is to run as root `touch /var/opt/NVMesh/mgr/update_management_id`. The **Management** should then proceed to start.

There is no need to stop any services for this operation.

## 14.4.5. Reset Factory Defaults

---



The following procedure will erase all data on the **NVMesh 2.5.2** cluster! This procedure is irreversible.

To reset the cluster to a clean configuration, perform the following actions:

1. Stop the cluster, see [Shutdown NVMesh](#)
2. On **Management Servers**, run `mongo management --eval 'db.dropDatabase()'`. This will delete all volume configurations and product history.
3. On **Targets**, run as root `rm -rf /var/opt/NVMesh/toma/*`. This will delete all volume state information on the **Targets**.
4. On **Clients**, run as root `rm /var/opt/NVMesh/mcs/CLIENT/CONFIGURATION`.

5. Start the cluster, see [Start NVMesh](#)

## 14.4.6. Rename Hostname

---

**NVMesh 2.5.2** identifies **Target** and **Client** nodes based on their hostname. Therefore, renaming a host that is configured as part of a cluster requires some administrative steps.

1. The first step is to have the node’s components identify themselves with the new hostname.
  - a. Use `systemctl restart nvmeshcm` to restart the communications service to management.
  - b. Use `systemctl status nvmeshcm` to validate it is running.
2. The second step is to remove the entries for the previous hostname from the **Clients** and **Targets** sections
  - a. This can be done immediately in the **Clients** section using the multi-select functionality and clicking the **Delete** button in the top left corner.
  - b. For **Targets**, the node will appear as a new **Target** and the NICs and *Drives* will now be associated with it. This typically takes a few seconds. Once this has happened, the entry with the previous hostname will have no NICs and *Drives* associated with it and can be deleted. This is done in the **Targets** section using the multi-select functionality and clicking the **Delete** button in the top left corner.

## 14.4.7. Upgrade NVMesh

---

It is possible to upgrade to **NVMesh 2.5.2**. The following sections outline the procedure required. Note that there is a difference when upgrading a cluster of **NVMesh 1.3** or earlier versus upgrading a cluster with **NVMesh 2.0** or later.

Before upgrading **Management Servers**, ensure that you are running the appropriate versions of MongoDB and NodeJS. See [Installing MongoDB and NodeJS](#) for more information on how to install.

To upgrade NodeJS, it is sufficient to follow the instructions for installing the required version.

To upgrade MongoDB from 3.6 used with previous versions of **NVMesh** to MongoDB 4.2, **it is imperative first to upgrade to 4.0 and then proceed to upgrade to 4.2.**

Follow the instructions at the following links. Make sure to review the release notes for information about compatibility settings as well. The key commands are also captured in [Installing MongoDB and NodeJS](#) for 4.2.

Operating System	Upgrade to MongoDB 4.0	Upgrade to MongoDB 4.2
Red Hat / CentOS	<a href="#">Mongo 4.0 for Red Hat</a>	<a href="#">Mongo 4.2 for Red Hat</a>
Ubuntu	<a href="#">Mongo 4.0 for Ubuntu</a>	<a href="#">Mongo 4.2 for Ubuntu</a>
	<a href="#">Release Notes for Mongo 4.0</a>	<a href="#">Release Notes for Mongo 4.2</a>

**Note:** after upgrading a **Management** instance, if changes had been made to `/etc/opt/NVMesh/management.js.conf`, a new file named `/etc/opt/NVMesh/management.js.conf.rpmnew` may be created. This file will include new fields that are critical for the proper function of **Management**. Therefore, compare the contents of the two files and ensure that the new fields exist also in `/etc/opt/NVMesh/management.js.conf` to ensure proper function. Alternatively, replace `/etc/opt/NVMesh/management.js.conf` with `/etc/opt/NVMesh/management.js.conf.rpmnew` and reinsert any adjustments made.

## 14.4.7.1. Upgrading from NVMesh 1.3 or older

It is possible to upgrade a cluster with **NVMesh 1.3.2** to **NVMesh 2.5.2**. To upgrade a cluster with a version prior to 1.3.2, contact [Excelero Technical Support](#) for help in upgrading to version 1.3.2 and then follow this procedure.

The upgrade is a warm upgrade, except for **Management**. It can be performed in the same manner as upgrading between 2.0 versions with the following additional instructions.

1. Update the **Management Servers** first.
  - a. Stop all **Management Servers**.
  - b. Update the **Management** packages, but do not restart the service yet.
  - c. Update NodeJS and Mongo, see [Upgrading](#).
  - d. Then, restart the **Management Servers** one by one.
2. The **Client** upgrade is warm, not hot, therefore restart the `nvmeshclient` service in the usual manner, using `systemctl restart nvmeshclient`.

**Note:** [Excelero Technical Support](#) can provide assistance using a technician tool for automating this process.

## 14.4.7.2. Upgrading from NVMesh 2.0 or newer

✿ Check the [Release Notes](#) for up-to-date information on upgrades.

Upgrading between minor versions of **NVMesh 2.x**, can be done using a warm upgrade for **Management Servers** and **Targets** and a hot upgrade for **Clients**.

Upgrading any component requires installing its new package and then restarting the service. For **Clients**, when following the hot upgrade procedure, the restart will leave the block devices attached and active and won't require unmounting file systems or stopping applications using the block devices.

✿ **Management Servers** should all be turned off as a safeguard during the upgrade of **Clients**

and **Targets**, including service restart.

## Upgrade Overview

1. Stop all **Management Servers**, upgrade them, but do not restart them.
2. Upgrade all **Targets** and **Clients**.
3. Verify that all **Targets** and **Clients** that are running have the new version.
4. Start one **Management** instance with the new version.
  - a. Verify that the system is online properly.
5. Start all other **Management** instances with the new version.

✿ Under rare circumstances, the **Client** may not restart properly after an upgrade. In this case, try again to start it after starting a **Management** instance.

## Upgrading Management Servers

Update the **nvmesh-management** package, as root:

- For Redhat/CentOS: `yum install nvmesh-management-<PACKAGE DETAILS>.rpm nvmesh-utils-<PACKAGE_DETAILS>.rpm`
- For Ubuntu: `dpkg -i nvmesh-management-<PACKAGE DETAILS>.deb nvmesh-utils-<PACKAGE_DETAILS>.deb --force-confold`
  - The *force-confold* flag instructs to keep the previous configuration during upgrades. Alternatively, press 'Y' when prompted whether to retain the old configuration.

Restart the service using `systemctl restart nvmeshmgr`.

## Upgrading Targets

Update the **nvmesh-core** package, as root:

- For Redhat/CentOS: `yum install nvmesh-core-<PACKAGE DETAILS>.rpm nvmesh-utils-<PACKAGE DETAILS>.rpm`
- For Ubuntu: `dpkg -i nvmesh-core-<PACKAGE DETAILS>.deb nvmesh-utils-<PACKAGE DETAILS>.deb --force-confold`
  - The *force-confold* flag instructs to keep the previous configuration during upgrades. Alternatively, press 'Y' when prompted whether to retain the old configuration.

Restart the services using `systemctl stop nvmeshclient ; sleep 1; systemctl start nvmeshtarget`. It is best to restart the **Client** and **Target** simultaneously.

If the **NVMe-oF** volume export functionality is utilized, the **nvmesh-nvmft** package should also be updated in

concert and the `nvmeshnvmft` service restarted as well.

## Upgrading *Clients*

Update the `nvmesh-core` package, as root:

- For Redhat/CentOS: `yum install nvmesh-core-<PACKAGE DETAILS>.rpm nvmesh-utils-<PACKAGE DETAILS>.rpm`
- For Ubuntu: `dpkg [ --force-confold ] -i nvmesh-core-<PACKAGE DETAILS>.deb nvmesh-utils-<PACKAGE DETAILS>.deb`
  - The *force-confold* flag instructs to keep the previous configuration during upgrades. Alternatively, press 'Y' if and when prompted whether to retain the old configuration. For some upgrades, you may be prompted.

### Hot-upgrade for *Clients*

It is possible to do a hot upgrade of *Clients* without requiring a detach from the volumes, which can be complex if there are mounted file systems or other applications use the block devices. To do this, first inform the *Client* that a hot upgrade is required. This will prepare the *Client* to restart its software to the new version by detaching a shim layer that will continue to maintain the block devices from the main *Client* software. I/O will become disabled until the restart is complete. Then, perform the restart. These steps are performed using the following steps:

1. `nvmesh_detach_volumes --client_shutdown --upgrade`
2. `systemctl restart nvmeshclient`

Running the second command only will restart the entire *Client* software stack including the shim layer (kernel module `nvmeiba`), but will require being able to detach all volumes.

If the *NVMe-oF* volume export functionality is utilized, the `nvmesh-nvmft` package should also be updated in concert and the `nvmeshnvmft` service restarted as well.

## 14.4.7.3. Upgrading NVMesh Management in Docker Container

---

To upgrade NVMesh management that is run as a Docker container perform the following steps:

1. Get the version of the currently running management container `docker exec -it <old mgmt container name> cat /opt/NVMesh/management/dbVersion`  
You should get some output like `2.2.0-40`
2. Stop the currently running management container `docker stop <old mgmt container name>`
3. Stop MongoDB service or container
4. Start the new management container. Please refer to [Deploying NVMesh Management in Containers](#)

for examples on starting NVMesh management containers.

The new container should fail to start since MongoDB is down – this is expected.

5. Edit the dbVersion file in the new management container and change the value to the old version.
 

```
docker exec -it <new mgmt container name> sh -c 'echo 2.2.0-40 > /opt/NVMesh/management/dbVersion'
```
6. Stop the management container
7. Start MongoDB service or container
8. Start the new management container. It should now properly start.

When running `docker logs <new mgmt container name>` you should see it running upgrade scripts with messages like `nvmeshmgr[85]: DEBUG Going to run upgrade script:`

```
2.4.0-2
```

## 14.4.8. Advanced Operations

---

### 14.4.8.1. Disabling Rebuild Operations

---

It is possible to disable and re-enable rebuild operations on a single volume or on all volumes from a single node.

It is highly unrecommended to do this over a long period of time, but this may be relevant for certain performance measurements.

Remember to re-enable them to prevent risk of data loss.

Disabling is temporary and will revert upon a detach and attach of a volume or a **Client** restart.

To disable for all volumes, run the following:

```
# Prevent new recoveries from starting
sudo bash -c 'echo -n "#*|ignore_toma_rcv=1" > /proc/nvmeibc/cli/cli';
#
# Stop currently running recoveries. TOMA will try to restart, but client will ignore it
sudo bash -c 'echo -n "#*|topo_flush" > /proc/nvmeibc/cli/cli';
#
echo Running: ignore_recoveries
#
# Easy to see the index of volume and its recovery disabled flag.
cat /proc/nvmeibc/volumes/*/status | grep -e 'itm=' -e 'short_id=';
```

Replace `#*` with `#<VOLUME_NAME>` to perform this for a single volume.

To re-enable recoveries set `ignore_toma_rcv` back to 0, as follows:

```
# Prevent new recoveries from starting
sudo bash -c 'echo -n "#*|ignore_toma_rcv=0" > /proc/nvmeibc/cli/cli';
#
# This will prompt TOMA to restart and client will begin perform the operation now
sudo bash -c 'echo -n "#*|topo_flush" > /proc/nvmeibc/cli/cli';
#
echo Stopping: ignore_recoveries
# Easy to see the index of volume and its recovery disabled flag.
cat /proc/nvmeibc/volumes/*/status | grep -e 'itm=' -e 'short_id=';
```

# 15. Configuration Limits

---

The following sections describe various system limits.

## 15.1. Client Limitations

---

Item	Limitation
Number of <i>Clients</i>	4,096
<i>Client</i> hostname	64 characters

## 15.2. Drive Limitations

---

Item	Limitation
Maximum number of <i>Drives</i> in an <i>NVMesh 2.5.2 Target</i>	60
Maximum number of <i>Drives</i> behind a single volume	256
Maximum number of <i>Drives</i> in an <i>NVMesh 2.5.2</i> cluster <b>per zone</b>	1,500
Maximum number of <i>Drive</i> segments in an <i>NVMesh 2.5.2</i> cluster <b>per zone</b>	100,000
Maximum volume stripe width in an <i>NVMesh 2.5.2</i> cluster <b>per zone</b>	300
Maximum number of concurrent <i>Drives</i> rebuilds per <i>Target</i>	2

## 15.3. Drive Class Limitations

---

Item	Limitation
<i>Drive Class</i> name	1,024 Unicode characters.
<i>Drive Class</i> description	1,024 Unicode characters.
Maximum number of <i>Drives</i> in a <i>Drive Class</i>	1,500
Maximum number of <i>Drive Classes</i> in an <i>NVMesh 2.5.2</i> cluster	5,000

## 15.4. Key Pair Limitations

---

Item	Limitation
------	------------

<i>Key Pair</i> name	1,024 Unicode characters.
<i>Key Pair</i> description	1,024 Unicode characters.

## 15.5. Networking Limitations

---

Item	Limitation
Maximum number of NICs per node	10
Maximum number of NICs per <i>NVMesh 2.5.2</i> cluster per zone	500
Maximum number of NIC ports per <i>Client</i> or <i>Target</i>	10

## 15.6. Target Limitations

---

Item	Limitation
Number of <i>Targets</i>	256 per zone
Maximum number of NVMe <i>Drives</i> per <i>Target</i>	60
<i>Target</i> hostname	32 characters.

## 15.7. Target Class Limitations

---

Item	Limitation
<i>Target Class</i> name	1,024 Unicode characters.
<i>Target Class</i> description	1,024 Unicode characters.
Number of <i>Target Classes</i>	5,000

## 15.8. User Limitations

---

Item	Limitation
User name	32 characters, uppercase and lowercase English letters, digits, and any of the special characters ._%+-
Password	6 to 32 Unicode characters

# 15.9. Volume Limitations

Item	Limitation
Volume name	24 characters, uppercase and lowercase English letters, digits, and any of the special characters _+ -=
Volume description field	1,000 Unicode characters.
Number of <i>MeshProtect 1</i> volumes *	8,192
Minimum volume size	1 GB
Maximum volume size	As large as the available capacity in the cluster.
Maximum number of <i>Clients</i> connected to a single volume	1,024

\* The limit applies to volumes that were allocated in one shot and which are implemented as a single drive per volume element. Otherwise, contact [Excelero Technical Support](#) for help in accounting for the number of volumes. For other volumes types, see the table below.

Volume Type	Ratio to MeshProtect 1	Variable for Formula
MeshProtect Concatenated	1/3	Nconcat
MeshProtect 0	1/3 * Volume Width	Nr0
MeshProtect 1	1	Nr1
MeshProtect 10	1 * Volume Width	Nr10
MeshProtect 60	1/2 * (Data + Parity)	Nec

$$N_{r1\_equivalent} = \frac{1}{3} * (N_{concat} + \sum_{MeshProtect0} \text{Volumes} * Vol_{width}) + N_{r1} + \sum_{MeshProtect10} \text{Volumes} * Vol_{width} + \frac{1}{2} * \sum_{MeshProtect60} \text{Volumes} * (Vol_{Data} \text{ Blocks} + Vol_{Parity} \text{ Blocks})$$

# 15.10. Volume Provisioning Group Limitations

Item	Limitation
<i>Volume Provisioning Group</i> name	1,024 Unicode characters.
<i>Volume Provisioning Group</i> description	1,024 Unicode characters.
Maximum number of <i>Volume Provisioning Groups</i>	5,000

## 15.11. Volume Security Group Limitations

---

Item	Limitation
<i>Volume Security Group</i> name	1,024 Unicode characters.
<i>Volume Security Group</i> description	1,024 Unicode characters.

# 16. Reference Information

## 16.1. SystemD Reference Table

### SystemD Action Results

- The start row lists all units that will be started upon a start command to the unit in the column header.
- The stop row lists all units that will be stopped upon a stop command to the unit in the column header.
- The restart row lists all units that will be restarted upon a restart command to the unit in the column header.
- The auto-restart row denotes whether the unit in the column header is restart automatically upon a failure.

Action / Unit	nvmeshclient	nvmeshtarget	nvmeshmgr	nvmeshtoma	nvmeshcm	nvmeshagent	nvmeshtrace	nvmesh...
<b>start</b>	nvmeshclient nvmeshcm nvmeshagent nvmeshtrace	nvmeshtarget nvmeshtoma nvmeshclient nvmeshcm nvmeshagent nvmeshtrace	nvmeshmgr mongod	nvmeshtarget nvmeshtoma nvmeshclient nvmeshcm nvmeshagent nvmeshtrace	nvmeshcm	nvmeshagent	nvmeshtrace	nvmesh...
<b>stop</b>	nvmeshtarget nvmeshtoma nvmeshclient nvmeshcm nvmeshagent nvmeshtrace	nvmeshtarget nvmeshtoma	nvmeshmgr	nvmeshtoma	nvmeshcm	nvmeshagent	nvmeshtrace	nvmesh...
<b>restart (1)</b>	nvmeshtarget nvmeshtoma nvmeshclient nvmeshcm nvmeshagent	nvmeshtarget nvmeshtoma	nvmeshmgr	nvmeshtoma	nvmeshcm	nvmeshagent	nvmeshtrace	nvmesh...
<b>auto-restart</b>	No	No	Yes	No	Yes	Yes	Yes	No

**Notes:**

(1) – It is possible to restart the **Client** (`nvmeshclient`) without detaching volumes or stopping services dependent upon currently attached volumes such as unmounting filesystems, using the following commands as root:

1. `nvmesh_detach_volumes --client_shutdown --upgrade`
2. `systemctl restart nvmeshclient`

# 17. Document Reference

## Typographical Conventions

Throughout this document, the following typographical conventions are followed:

Style	Meaning
<b>bold text</b>	The name of an Excelero software component or technology
text	A file name, command or configuration text that can be utilized in a Linux terminal/shell, file or as a URL
<i>term in italics</i>	Generally, a term being used in specific relation to an element in <b>NVMesh 2.5.2</b>

## Definitions

Throughout this document, these terms have the following meanings:

Term	Definition
<b>Management</b> or <b>Management Servers</b>	The server(s), or OS image(s) running the <b>management module</b> software
<b>Target</b>	A physical server containing one or more NVMe SSDs running the <b>storage target module</b>
<b>Client</b>	An OS image instance running the <b>block storage client</b> software
Volume	A logical block device defined in <b>NVMesh 2.5.2</b> for block storage consumption by <b>Clients</b>
<b>RDDA</b>	Remote Direct Drive Access. Excelero's patented low-latency and CPU bypass transport technology.
<b>TOMA</b>	<b>Topology Manager</b> . The <b>storage target module</b> component that handles error detection and volume rebuild activities.