NVMe-oF and Swordfish

Version: 1.2.8

ABSTRACT:

This paper provides a deep dive into the NVMe-oF configurations, and more specifically, how these are represented in both the Swordfish client model and API. It will also focus on the concepts of logical devices, called exported resources, and how these are represented, allocated and managed, as these are represented differently for NVMe-oF devices than for other types of storage devices modeled in Swordfish

SNIA Approved Publication

Table of Contents

List of Tables

Table of Figures

USAGE

that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of Storage Networking Industry Association nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

> THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

DISCLAIMER

The information contained in this publication is subject to change without notice. SNIA makes no warranty of any kind with regard to this publication, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The SNIA shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use.

Suggestions for revisions should be directed to http://www.snia.org/feedback/.

Current Revision

SNIA is actively engaged in expanding and refining the Swordfish documentation. The most current revision can be found on the SNIA web site at https://www.snia.org/tech_activities/standards/curr_standards/swordfish.

Contact SNIA

Current SNIA practice is to make updates and other information available through their web site at

http://www.snia.org.

FEEDBACK AND INTERPRETATIONS

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent via the SNIA Feedback Portal at http://www.snia.org/feedback/ or by mail to SNIA, 5201 Great America Parkway, Suite 320, Santa Clara, CA 95054, USA.

INTENDED AUDIENCE

This document is intended for use by individuals and companies engaged in storage management.

VERSIONING POLICY

This document is versioned material. Versioned material shall have a three-level revision identifier, comprised of a version number 'v', a release number 'r' and an errata number 'e'. Future publications of this document are subject to specific constraints on the scope of change that is permissible from one revision to the next and the degree of interoperability and backward compatibility that should be assumed between products designed to this standard. This versioning policy applies to all SNIA Swordfish versioned materials.

Version Number: Versioned material having version number 'v' shall be backwards compatible with all of revisions of that material that have the same version number 'v'. There is no assurance of interoperability or backward compatibility between revisions of a versioned material with different version numbers.

Release Number: Versioned material with a version number 'v' and release number 'r' shall be backwards compatible with previous revisions of the material with the same version number, and a lower release number. A minor revision represents a technical change to existing content or an adjustment to the scope of the versioned material. Each minor revision causes the release number to be increased by one.

Errata Number: Versioned material having version number 'v', a release number 'r', and an errata number 'e' should be backwards compatible with previous revisions of the material with the same version number and release number ("errata versions"). An errata revision of versioned material is limited to minor corrections or clarifications of existing versioned material. An errata revision may be backwards incompatible, if the incompatibility is necessary for correct operation of implementations of the versioned material.

Revision History

The evolution of this document is summarized in Table 1}.

Table 1: Revision History

Date	Rev	Notes
22 January 2024	1.2.6	Initial version; Release as Working Draft
9 April 2024	1.2.6	Release as SNIA Standard
21 May 2024	1.2.7	Release as Working Draft
		Updated diagrams for clarity.
28 January 2025	1.2.8	Release as Working Draft
		- Added more details for exporting resourcea dna managing exported resources.
		- Correct formatting errors in document.
22 May 2025	1.2.8	Release as SNIA Standard

About SNIA

SNIA is a not-for-profit global organization made up of corporations, universities, startups, and individuals. The members collaborate to develop and promote vendor-neutral architectures, standards, and education for management, movement, and security for technologies related to handling and optimizing data. SNIA focuses on the transport, storage, acceleration, format, protection, and optimization of infrastructure for data. Learn more at www.snia.org.

Acknowledgements

The SNIA Scalable Storage Management Technical Work Group, which developed and reviewed this white paper, would like to recognize the significant contributions made by the following members listed in Table Table 2.

Table 2: Contributors

Member	Representatives (* – prior employer)
Intel Corporation	Richelle Ahlvers
	Phil Cayton

1 Background

What is NVMe?

NVM Express® (NVMe) is a standard interface and protocol library developed to fully realize the benefits of Non-Volatile Memory (NVM) by accelerating access to Non-Volatile Memory devices (e.g., SSDs).

The NVMe® specification family defines how hosts communicate with non-volatile memory either directly, via the PCIe interface, or indirectly, through one or more of the supported NVMe fabric transports (e.g., RDMA, Fibre Channel, TCP). Indirectly accessing NVMe devices over fabrics extends the low-latency, efficient, NVMe storage protocol to provide scale-out access to, and sharing of, storage from remote storage systems (e.g., storage servers, storage appliances). NVMe maintains the same architecture and software of the NVMe protocol, providing the benefits of NVMe regardless of the fabric type or the type of non-volatile memory used in the storage target or appliance.

As NVMe supports every major storage interconnect, it has unified client, cloud, edge, and enterprise storage around a common command set and architecture. NVMe has become the language of storage for both data center servers and client devices.

The NVMe family of specifications is maintained and managed by the NVM Express non-profit industry association.

What is SNIA Swordfish?

The SNIA Swordfish Specification provides a standards-based REST interface for clients to manage multiple NVMe and NVMe-oF devices, either in a single system, or across multiple systems. As part of the Swordfish suite of documentation, the Swordfish NVMe Mapping Guide provides a translation between the Swordfish specification and the NVMe specification for implementers, ensuring that clients have consistency across implementations for the various types of NVMe devices.

2 NVMe Fundamentals

2.1 NVMe Networks and Transports

To connect NVMe devices, there is always a network between the initiator (host) and the target (device). For simple connections, this is a point-to-point style network using PCIe. For more complex configurations, different types of fabric transports are used, such as Ethernet, Fibre Channel, and InfiniBand. Figure 1 provides a high-level summary of common NVMe transports and how they might be used.
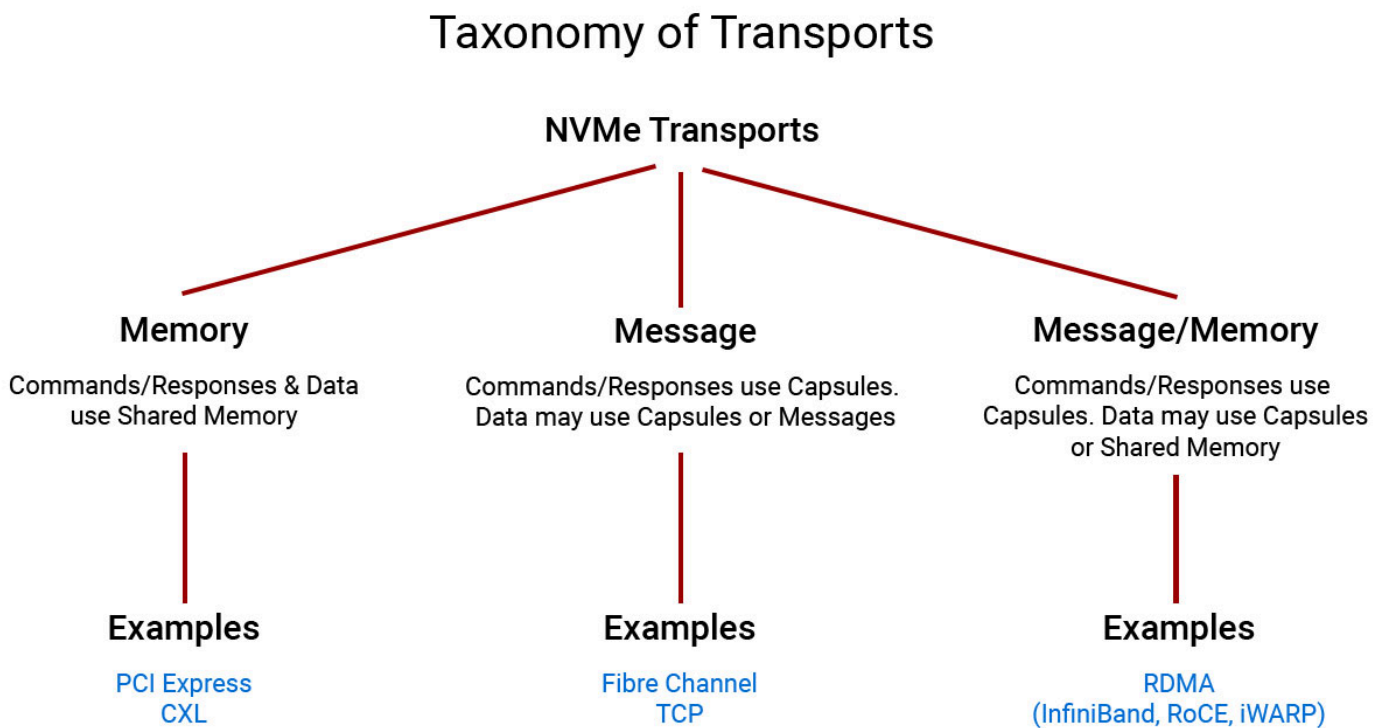

Figure 1: High-level Taxonomy of Transport

2.2 Underlying NVM Resources

2.2.1 Underlying Namespaces

In NVMe, An Underlying Namespace is a namespace (i.e., a formatted quantity of non-volatile memory) on an Underlying Subsystem that is available and accessible by the storage server. Underlying namespaces are identified by a Underlying Controller ID, Underlying NVM Subsystem, and Underlying Namespace ID combination.

These Underlying Namespaces may be resident on NVM devices in the storage server and accessible via physical functions or attached to Exported NVMe Subsystems resident on other storage servers or storage devices and accessible via virtual functions.

Storage server operating systems may maintain an 'Underlying Namespace List' with all Underlying Namespaces that may be exported over fabrics (reference Figure Figure 2, note ①)

2.2.2 Underlying Ports

An Underlying Port represents a fabric connector (e.g., TCP, InfiniBand, Fibre Channel) that may be used to export an NVM Subsystem. The Underlying Port has a Port ID, which is an identifier associated with an NVM subsystem port. This port is included in the List of ports that may be used to export from the NVM subsystem. As with Underlying Namespaces, a storage server operating system may maintain an 'Underlying Ports List' with all fabric ports that may be used to export an NVM Subsystem (Figure 2 note ②).
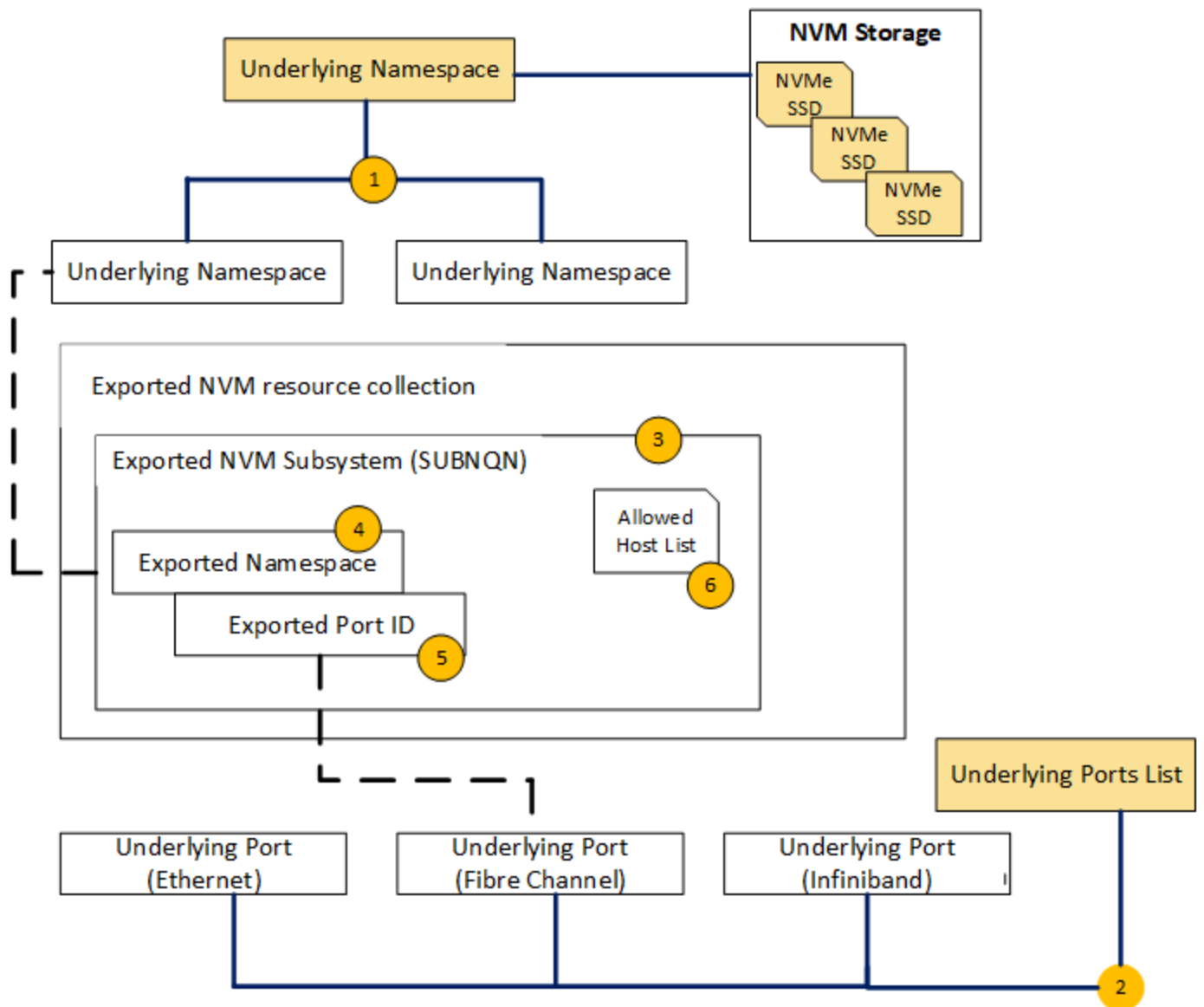
Figure 2: Exported NVM Namespaces and Ports

2.3 Exporting Underlying Namespaces over Fabrics

Prior to configuring exported NVM resources and exposing them for access, an administrative entity (e.g., administrator, resource manager, orchestrator, administration console, centralized configuration manager) must first determine what Underlying resources (e.g., namespaces, ports) are available; this may be through a-priori knowledge or by using NVMe Identify commands.

An Exported NVM Subsystem is a logical NVM subsystem contains zero or more Exported Namespaces, zero or more controllers, zero or more Exported Ports, a flag indicating the access mode (either: 'allow-all' or 'restricted access'), and an Allowed Host List which is a list of hosts (identified by Host NQN and Host Identifier) that are granted to the Exported NVM Subsystem via an Exported Port.

An Underlying Namespace may be made available to remote consumers over fabrics via an Exported Namespace. An Exported Namespace is a virtual representation of an Underlying Namespace that is formed by creating an Exported Namespace ID and associating the Exported Namespace ID with an Underlying Namespace. The Exported Namespace must then be associated with an Exported NVM Subsystem which has been configured with the fabric transport(s) on which the Exported NVM Subsystem and associated Exported Namespace(s) may are to be accessible.

This section will detail the components and steps to make NVM namespaces available over fabrics.

2.3.1 Creating Exported Ports

An Exported Port is a port used to export an NVM Subsystem over a specific underlying fabrics transport (e.g., TCP, InfiniBand, Fibre Channel), and represented by an Exported Port ID (Figure 2 ⑤). Exported Ports are created through an NVMe administrative command.

2.3.2 Creating Exported NVM Subsystems

Exported NVM Subsystems (Figure 2 note ③) are created through an NVMe administrative command.

Access to an Exported NVM Subsystem and all Exported NVM Namespaces associated with that Exported NVM Subsystem is managed by configuring the Exported NVM Subsystem for either 'Unrestricted Access' (i.e., the Exported NVM Subsystem may be accessed by any Host), or 'Restricted Access' (i.e., the Exported NVM Subsystem may only be accessed by Host NQNs (NVMe Qualified Name) present in the Exported NVM Subsystem's 'Allowed Host List').

Administrators may choose to create an Exported NVM Subsystem with either unrestricted access, or restricted access. The default is to create Exported NVM Subsystems with 'unrestricted' access, however the administrative entity may choose to create the Exported NVM Subsystems with access restricted to hosts whose host NQN is present in the Exported NVM Subsystem's Allowed Host List.

Note that on creation of an Exported NVM Subsystem, the Exported NVM Subsystem's Allowed Host List will be empty, and for Exported NVM Subsystems created with 'restricted' access, the Exported NVM Subsystem's Allowed Host List must be populated to enable any host to access to the Exported NVM Subsystem and associated Exported Namespaces.

In order for allowed hosts to access Exported NVM Subsystems over fabrics, each Exported NVM Subsystem must be associated with at least one Exported Port. Exported NVM Subsystems are associated with an Exported Port through an NVMe administrative command.

2.3.3 Exporting Namespaces

To export an Underlying Namespace over fabrics it is necessary to associate a new or existing Exported Namespace ID with an Underlying Namespace, associate the Exported Namespace ID with an Exported

NVM Subsystem, and attach the Exported Namespace to a controller to make an Exported Namespace an active namespace.

Note: if Namespace Attribute Notices are enabled, the controller(s) newly attached to the Exported Namespace shall report a Namespace Attribute Changed asynchronous event to the host.

Once an Exported Namespace ID has been associated with an Underlying Namespace and then associated with Exported NVM Subsystem which in turn has been associated with an Exported Port ID, then the namespace may be assessed over the configured network protocol.

2.3.4 Summary

Exporting a namespace over fabrics may thus be achieved by:

- optionally retrieving the list of Underlying NVM Namespaces (if available) – Figure 2, note [1]. The list of Underlying NVM Namespaces, if supported by the system, constitutes the set of NVM namespaces that may be exported over fabrics via an Exported NVM Subsystem.
- optionally retrieving the list of Underlying Ports (if available) – Figure 2 note [2]. The list of Underlying Ports, if supported by the system, constitutes the set of fabric transports that may be associated with an Exported NVM Subsystem to enable host access to the Exported NVM Subsystem.
- creating an Exported NVM Subsystem and optionally assigning access control policies for the Exported NVM Subsystem Figure 2 note [4].
- associating one or more Underlying Namespaces with an Exported NVM Subsystem.
- associating one or more transports with the Exported NVM Subsystem to enable remote access to the Exported NVM Subsystem – Figure 2 [5].
- optionally assigning access control policies for the Exported NVM Subsystem by altering Exported NVM Subsystem from its initial or current access control mode (i.e., unrestricted access or restricted access).
- optionally granting or revoking access to the Exported NVM Subsystem by adding or removing Host NQNs from the Exported NVM Subsystem's Allowed Host List.

2.4 Managing and Deleting Exported NVM Subsystems

2.4.1 Managing Host Access to Exported NVM Subsystems

The access mode of an Exported NVM Subsystem may be changed at any time by issuing an NVMe administrative command to toggle the Exported NVM Subsystem 'Restricted Access' configuration bit between 'Unrestricted Access' mode and 'Restricted Access' mode. If Exported NVM Subsystem access is changed from 'Unrestricted Access' to 'Restricted Access', then any connected host not in the Allowed Host

List associated to the specified Exported NVM Subsystem shall be disconnected from all Exported Namespaces in the Exported NVM Subsystem.

Furthermore, the Host Access List for a given Exported NVM Subsystem may be modified to grant or revoke access by specific Hosts to the Exported NVMe Subsystem on specific Exported Ports.

2.4.2 Deleting Exported NVM Namespaces

To delete an Exported Namespace, Host software performs the following ordered actions:

1. detach the namespace from all controllers by issuing the Namespace Attachment command specifying the Controller Detach operation to detach the specified Exported Namespace from one or more controllers; if Namespace Attribute Notices are enabled, the controllers that were detached from the Exported Namespace report a Namespace Attribute Changed asynchronous event to the affected Host.
2. disassociate the Exported Namespace from the Exported NVM Subsystem through the appropriate NVMe administrative command.
3. delete the Exported Namespace through the appropriate NVMe administrative command.

2.4.3 Deleting Exported NVM Subsystems

To delete an Exported NVM Subsystem Host software performs the following ordered actions:

1. close all host connections to the Exported NVM Subsystem through the appropriate NVMe administrative command.
2. disassociate all Exported Namespaces from the Exported NVM Subsystem through the appropriate NVMe administrative command.
3. disassociate all Exported Port IDs from the Exported NVM Subsystem through the appropriate NVMe administrative command.
4. delete the Exported NVM Subsystem through the appropriate NVMe administrative command.

2.4.4 Unavailable Namespaces

If an Exported Namespace is detached from a controller in the Exported NVM Subsystem, then the Namespace Identifier that referred to that namespace becomes an inactive Namespace Identifier on that controller.

If an Exported NVM Subsystem Exports an Underlying Namespace that becomes unavailable (e.g., detached, deleted) or is affected by Capacity Management e.g., Endurance Group deletion, NVM Set deletion then:

- Previously submitted but uncompleted or subsequently submitted commands to the Exported Namespace are handled by the controller as if they were issued to an inactive NSID.
- If Namespace Attribute Notices are enabled controllers affected by Underlying Namespace

changes report a Namespace Attribute Changed asynchronous event to the host.

If an Underlying Namespace is disassociated from the Exported NVM Subsystem, then the Namespace Identifier of the Exported Namespace that referred to that namespace becomes an unallocated Namespace Identifier and is not available to any controller in the Exported NVM Subsystem. Furthermore, previously submitted but uncompleted or subsequently submitted commands to the namespace that is detached from a controller and/or disassociated from the Exported NVM Subsystem, are handled by the controller as if they were issued to an inactive Namespace Identifier.

2.4.5 Effects of Sanitizing Underlying NVM Subsystems on Exported Namespaces

Performing a sanitize operation on an Underlying NVM Subsystem sanitizes user data in all Underlying Namespaces contained in that NVM subsystem, including any Underlying Namespace that is associated with an Exported Namespace in any Exported NVM Subsystem.

If an Exported NVM Subsystem contains an Exported Namespace that is associated with an Underlying Namespace in an Underlying NVM Subsystem where a sanitize operation is in progress or the most recent sanitize operation has failed and successful recovery from the failed sanitize operation has not occurred, then, while that condition exists, that Exported NVM Subsystem shall enforce the I/O command sanitize operation restrictions on I/O commands that specify that Exported Namespace and may enforce additional sanitize operation restrictions.

3 Managing NVMe resources with Swordfish

Swordfish abstracts underlying exported NVM storage and fabric resources as specified in the NVMe 2.0 family of specifications to enable scalable storage management via RESTful interfaces. Figure 3 illustrates the Swordfish representation of NVMe and fabrics resources.
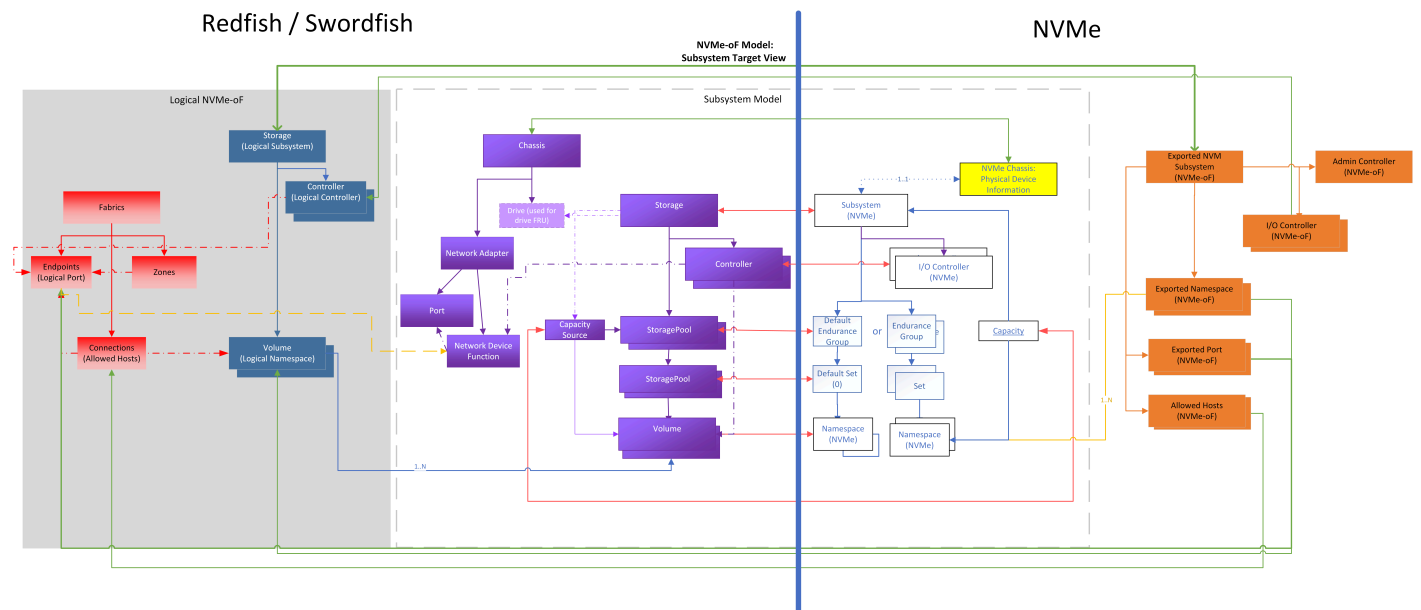

Figure 3: NVMe modelling in Swordfish

In the Swordfish representation of NVMe and fabrics resources (Figure 3) the white boxes represent the Underlying NVMe resources; the orange boxes represent Logical (Exported) representations of NVMe resources. As noted in the diagram, there is a largely 1:1 mapping in the Swordfish representation of the NVMe resources:

- The logical NVM subsystem is modeled as a Storage instance.
- A logical NVM ontroller is modeled as a StorageController instance, which is contained by a Storage instance.
- A logical amespace is modeled as a Volume instance, which is contained by a Storage instance, and related to StorageController instances.
- NVM Logical Ports are modeled as Endpoints within a Fabric instance.
- Allowed Hosts are modeled as Connections within a Fabric instance.

NVMe-oF systems can span a wide range of instantiations. They can encompass from simple, static configurations to highly complex, dynamic configurations with multiple components and interconnections across different topologies. Further, they support the creation and deletion of resources at multiple levels within the system.
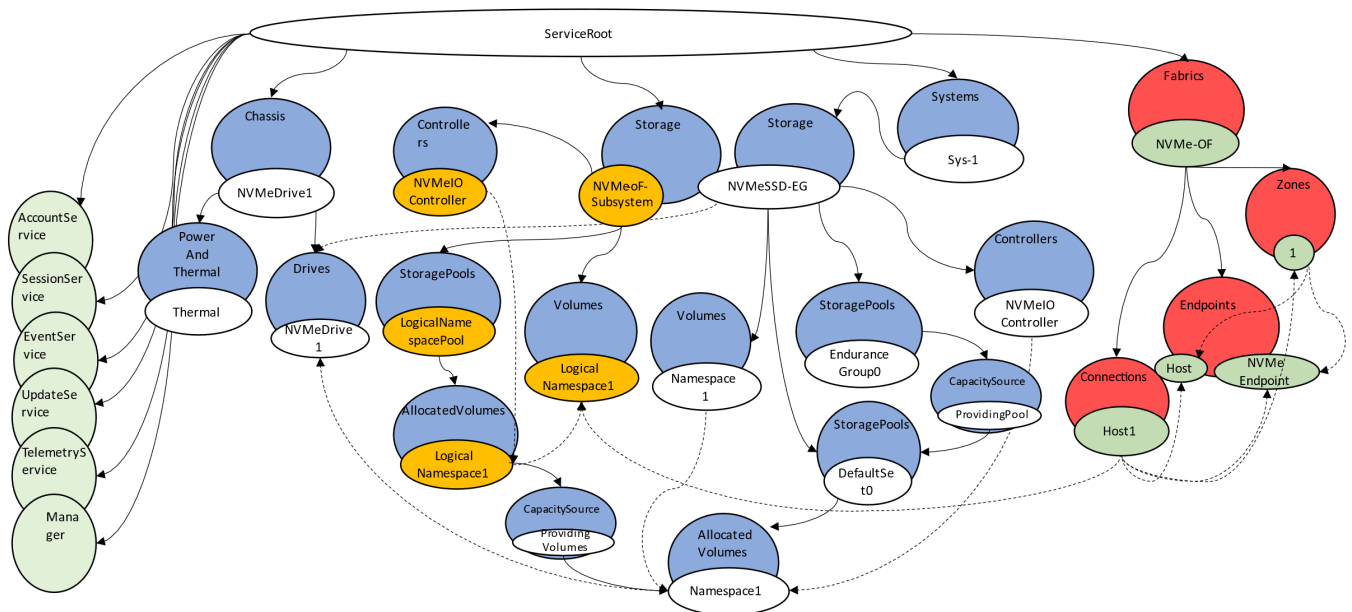

Figure 4: Logical NVMe-oF Instance

Figure 4 shows an instance of a simple logical NVMe-oF system construct. The logical NVMe-oF namespace (represented in orange) is created from a single underlying namespace.

Note that the logical controller is separate from the underlying resources used to create the logical namespace; it is not constructed as simply a pass-through for the logical controller on the underlying resources. This is a key point for the Swordfish modeling, as this follows the traditional storage pool / storage aggregation behavior. Logical namespaces are the linkage between the underlying model, while the remaining logical entities do not pull from the underlying components.

The representation in Figure 4 shows only the logical NVMe-oF. To be complete, we need the underlying transport as well, as shown in Figure 5.
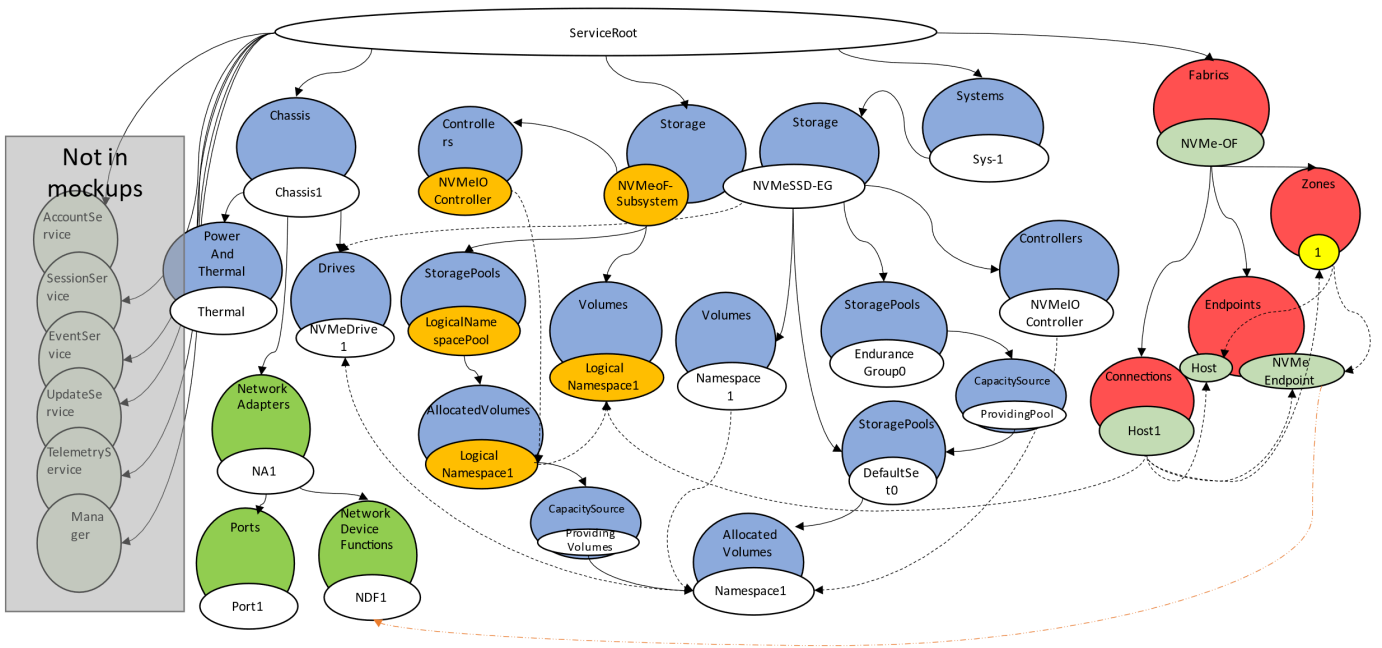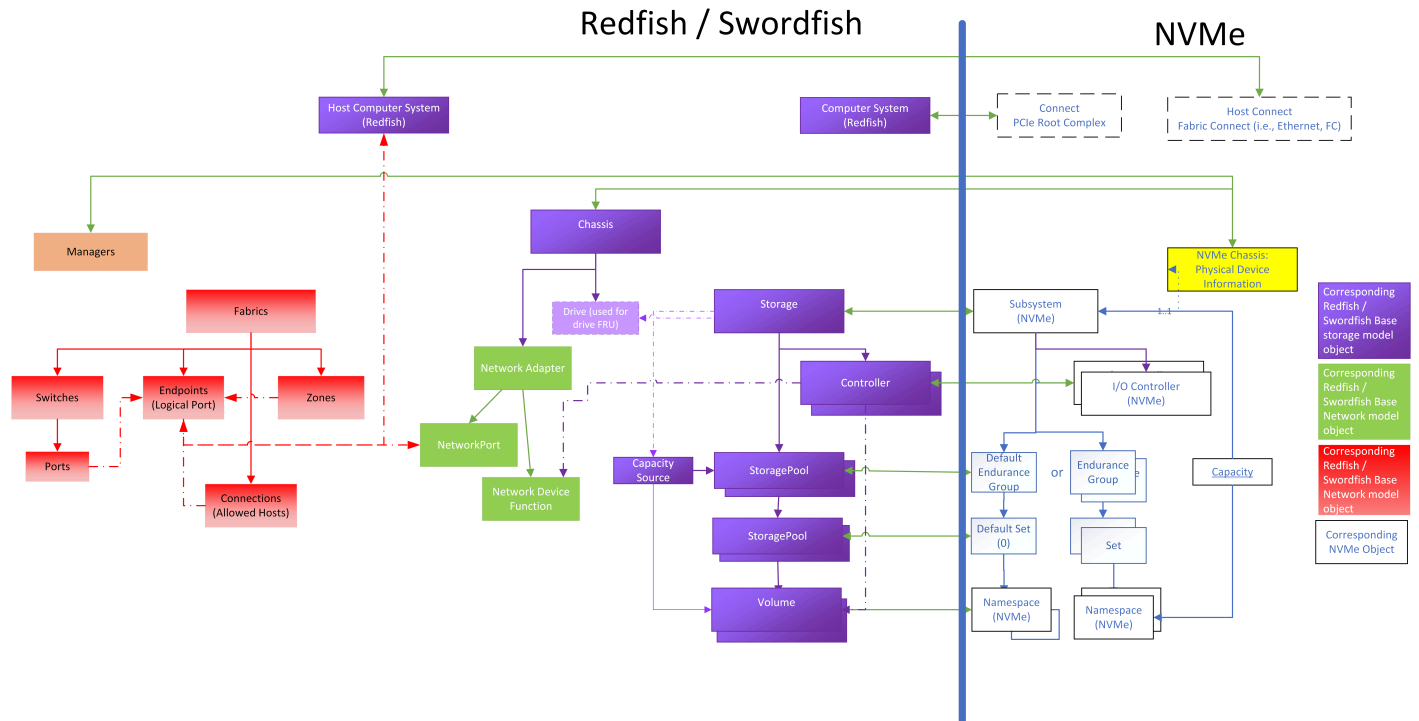
Figure 5: Logical NVMe-oF with Network Connection

The object model in Figure 6 shows the breakdown and grouping of the various objects within the overall model.

Figure 6: Redfish/Swordfish NVMe Model Including Network Objects

The above representations show an exported NVMe-oF instance comprised of a single SSD instance. However, this relationship can be a many to many (N:M) relationship, as in Figure 7 where a single logical namespace is created from two (2) underlying namespaces (a 1:2 configuration).

Correspondingly, as the underlying capacity was put into a storage pool, it could be allocated into any number of namespaces.

Figure 7: NVMe-oF Exported Logical Namespace from 2 SSDs

As demonstrated above, Swordfish can not only represent the advanced concepts of NVME-oF, but does so in a way consistent with the object model representations used for other storage and fabric representations, while still reflecting and exposing the unique attributes of NVMe-oF exported logical systems.