
# NVMe-oF and Swordfish

## Version: 1.2.6

**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 White Paper

*This document has been released and approved by the SNIA. The SNIA believes that the ideas, methodologies, and technologies described in this document accurately represent the SNIA goals and are appropriate for widespread distribution. Suggestion for revision should be directed to http://www.snia.org/feedback/.*

*Last Updated: 22 January 2024*

# Contents

## USAGE

Copyright (c) 2024 SNIA. All rights reserved. All other trademarks or registered trademarks are the property of their respective owners.

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

1. Any text, diagram, chart, table or definition reproduced must be reproduced in its entirety with no alteration, and,
2. Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced must acknowledge the SNIA copyright on that material, and must credit SNIA for granting permission for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, or any portion thereof, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by emailing tcmd@snia.org. Please include the identity of the requesting individual and/or company and a brief description of the purpose, nature, and scope of the requested use.

All code fragments, scripts, data tables, and sample code in this SNIA document are made available under the following license:

BSD 3-Clause Software License

Copyright (c) 2024, SNIA.

Redistribution and use in source and binary forms, with or without modification, are permitted provided 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 SNIA 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. The 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/.

**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 |

## 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 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 or 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.

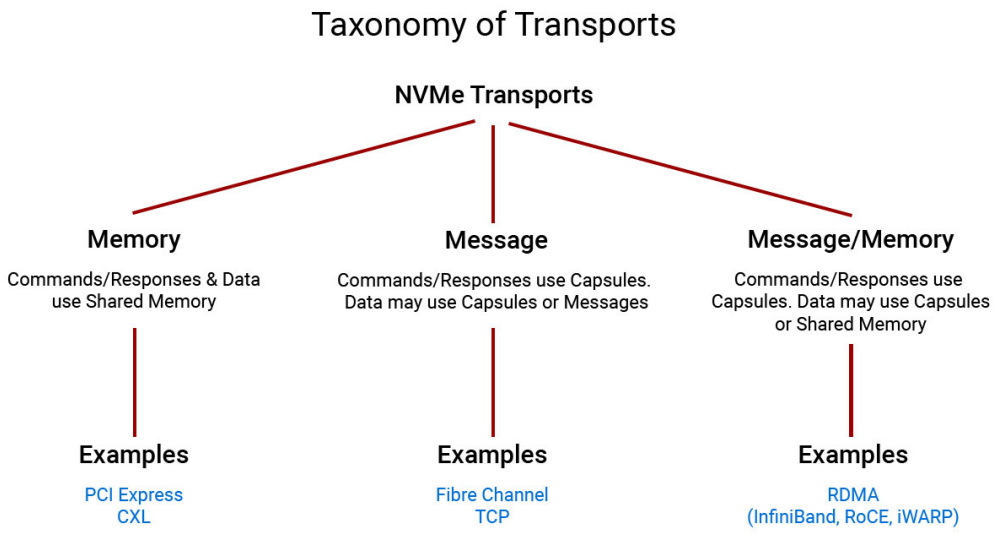



**Figure 1:** High-level Taxonomy of Transport

## 2.2 Underlying NVM Resources

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. These Underlying Namespaces may be resident on NVM devices in the storage server and accessible via physical functions or attached to Exported NVMe-oF 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 2, note 1)

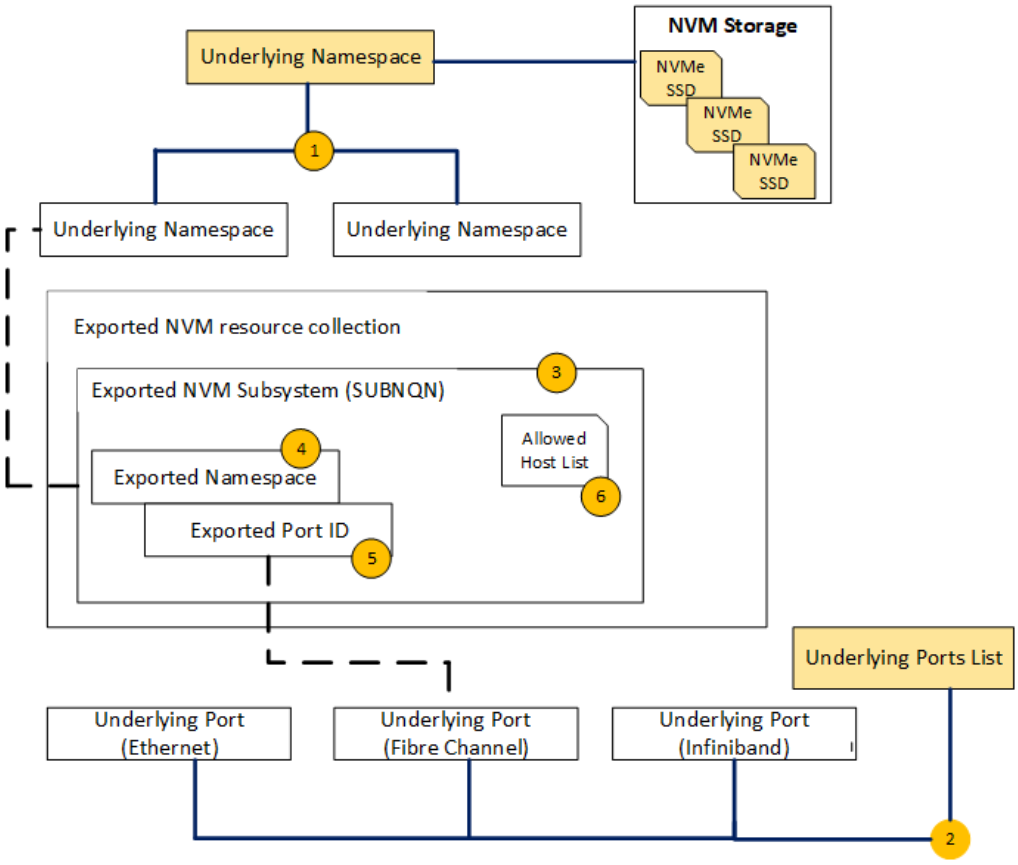



**Figure 2:** Exported NVM Namespaces and Ports

An Exported Namespace is a virtual representation of an Underlying Namespace that is formed by creating a new Exported Namespace ID and associating the Exported Namespace ID with an Underlying Namespace (identified by a Underlying Controller ID, Underlying NVM Subsystem, and Underlying Namespace ID combination),

An Exported Port is a port used to export an NVM Subsystem over a specific fabrics transport (e.g., TCP, InfiniBand, Fibre Channel), and represented by an Exported Port ID. 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).

## 2.3 Exporting Underlying NVM Namespaces over Fabrics

An Underlying Namespace may be made available to remote consumers over fabrics creating an Exported Namespace ID, associating the Exported Namespace ID with the Underlying Namespace, and associating the Exported Namespace with an Exported

NVM Subsystem which has been configured with the fabric transport(s) on which the Exported NVM Subsystem is to be accessible.

Access to Exported NVM Subsystems is managed first 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”). The access mode of an Exported NVM Subsystem may be changed at any time by toggling the “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.

Administrative commands described in the NVMe Specification enable:

- retrieving the list of Underlying Namespaces spanning all NVM subsystems that are accessible through either a virtual function or a virtual function (if the OS maintains the list)
- retrieving the list of Underlying Ports that may be used to export NVMe over Fabrics subsystems (if the OS maintains the list)
- creating an Exported NVM Subsystem, including setting the initial access mode for that Exported NVM Subsystem to either unrestricted access (i.e., the Exported NVM Subsystem may be accessed by any remote NVMe Host) or restricted access (i.e., the Exported NVM Subsystem may only be accessed by Hosts who’s NQN is present in the Exported NVM Subsystem’s Allowed Host List)
- managing Exported NVM Subsystems, including deleting an Exported NVM Subsystem, changing the access mode of an Exported NVM Subsystem, and modifying the Exported NVM Subsystem’s Allowed Host List by granting/revoking access to the Exported NVM Subsystem by specific NVMe Hosts (if the Exported NVM Subsystem is configured for restricted access)
- managing association of a configured for access through an Exported NVM Subsystem (i.e., by associating/disassociating a specific Underlying Namespace with a specific Underlying NVM Subsystem)
- managing associations of Exported Ports with Exported NVM Subsystems, and
- clearing the storage system Exported NVM resource configuration by deleting all Exported NVM resource configuration information and removing all Exported NVM Resources (i.e., Exported NVM Subsystems, Exported NVM

Namespaces, and Exported Ports).

## 2.4 Creating and Managing Exported NVM Subsystems

Exported NVM Subsystems (Firgure 2, note 3) are created by default with unrestricted access, however the administrative entity may choose to create the Exported NVM Subsystems with access restricted to hosts who's 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 must be populated to enable host access to the Exported NVM Subsystem configured for restricted access.

Exporting a namespace over an NVMe ransport may 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 transports that may be associated with an Exported NVM Subsystem to enable host access to the Exported NVM Subsystem.
- associating one or more Underlying Namespaces with an Exported NVM Subsystem – Figure 2, note 4
- 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.

Clearing Exported NVM Subsystem configuration may be achieved through either individual removal of specific Exported NVM Subsystems or through removing all NVM subsystems exported on the storage server/appliance.

# 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.

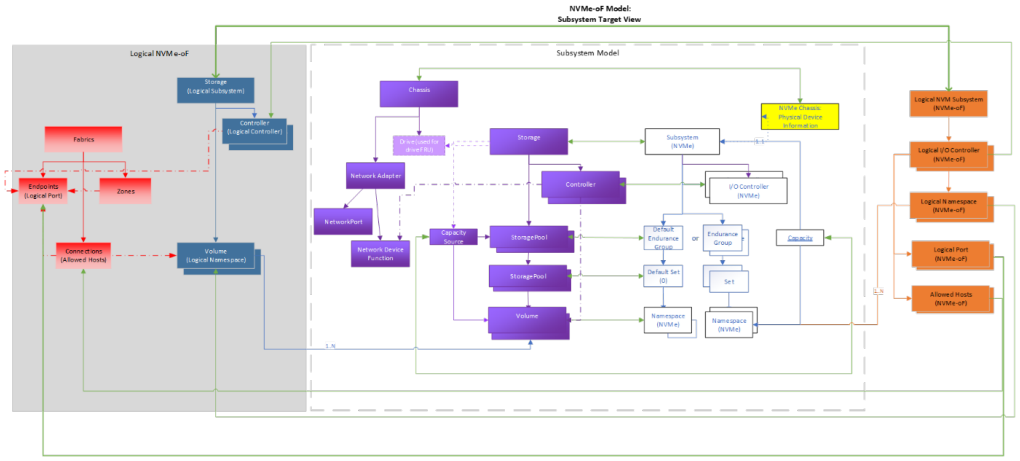



**Figure 3:** NVMe modelling in Swordfish

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.

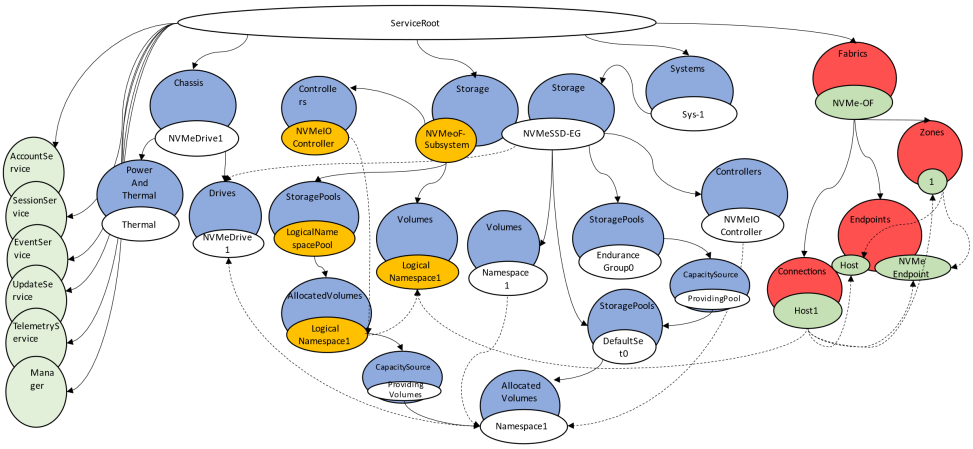



**Figure 4:** Logical NVMe-oF Instance

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.

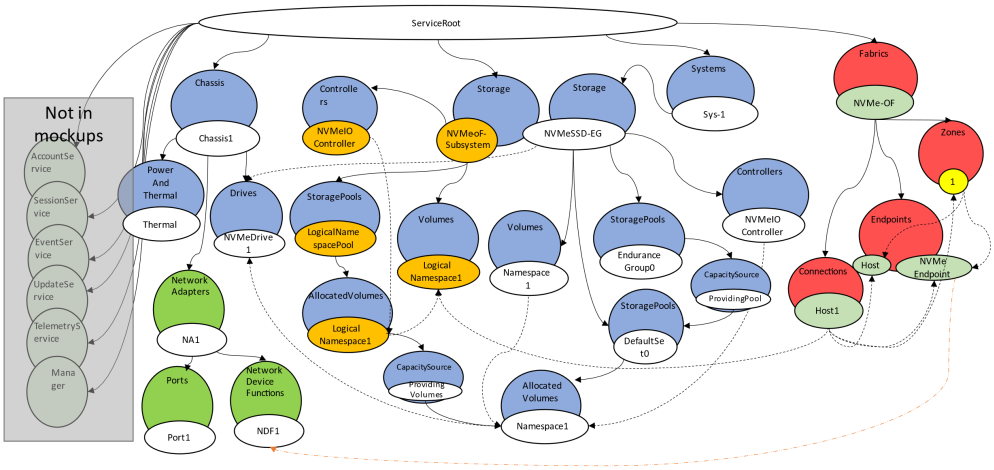



**Figure 5:** Logical NVMe-oF with Network Connection

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

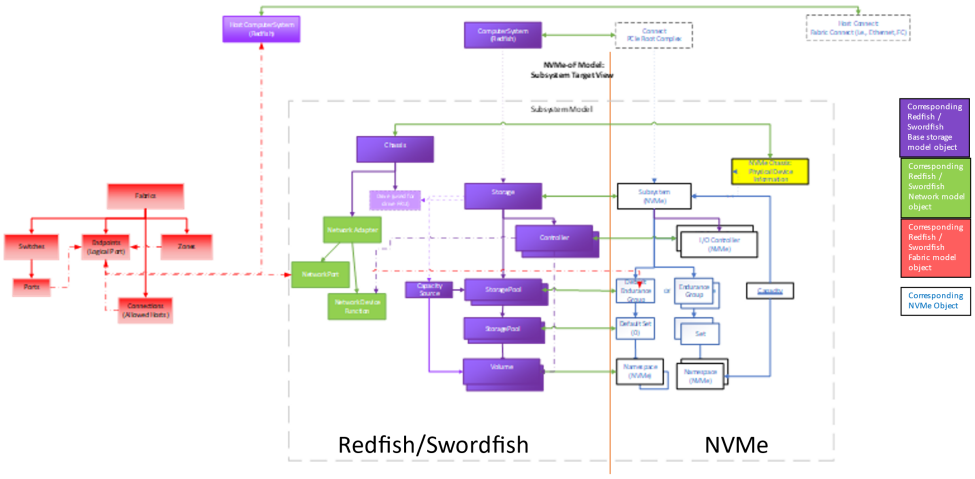
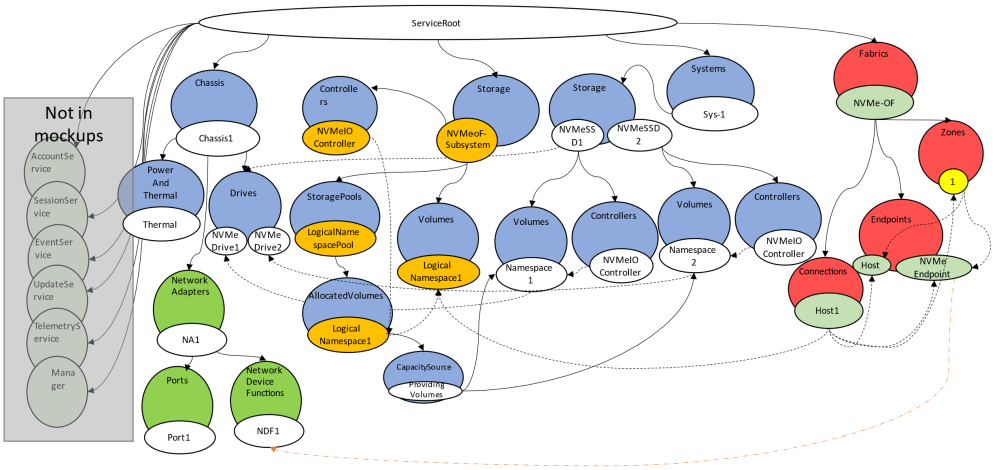



**Figure 6:** Redfish/Swordfish NVMe Model Including Network Objects

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 **??**, 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.




As demonstrated above, Swordfish can not only represent the advanced concepts of NMEoF, 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 NVMeoF exported logical systems.