Working Draft: NVMe to RF/SF Model Mapping

May 2020

Richelle Ahlvers, Broadcom

SSM TWG Chair

Working Draft 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 specification, 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 of this specification.

Feedback

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

TWG Status Disclaimer

- This proposal is a snapshot of work in progress by the SNIA Scalable Storage Management TWG, developed by the NVMe Task Force.
- Contents are subject to change at any time.

Related Materials

- This overview is released in conjunction with:
- The Swordfish v1.2.0 Specification and bundle release
  - Includes corresponding schema referenced in this presentation including, but not limited to: Volume, StoragePool, NVMeDomain
- NVMe and NVMe-oF mockups published on swordfishmockups.com

- The DMTF's Redfish Forum Work-in-Progress June 2020 release of DSP-IS0014 (v0.95)
  - Contains the schema updates referenced in this presentation for:
  - Storage, StorageController, Connection, Fabric, Resource, Protocol, Chassis, StorageControllerCollection, ConnectionCollection

Fundamental Design Assertions

- There shall be a unified model across all types of NVMe devices
  - There shall not be a different model for “drives” vs other types of NVMe devices
- The model will cover an appropriate level of abstraction for all types of NVMe devices based on modeling and mockups reflected in the documented permutations (e.g., from simple drives through to complex fabric virtual systems)
  - Simple NVMe drives; complex NVMe drives; JBOFs/EBOFs; Arrays/RBOFs
- The logical model for NVMe-oF shall leverage the NVMe Subsystem model
  - Logical subsystems, controllers, and namespaces are the same objects with the same relationships as in the NVMe Subystem Model. Unneeded objects are not instantiated (e.g., Endurance Groups, sets)

Overall NVMe Subsystem Model

- Model reflects a unified view of all NVMe device types.
- Devices will instantiate an appropriate subset of the model

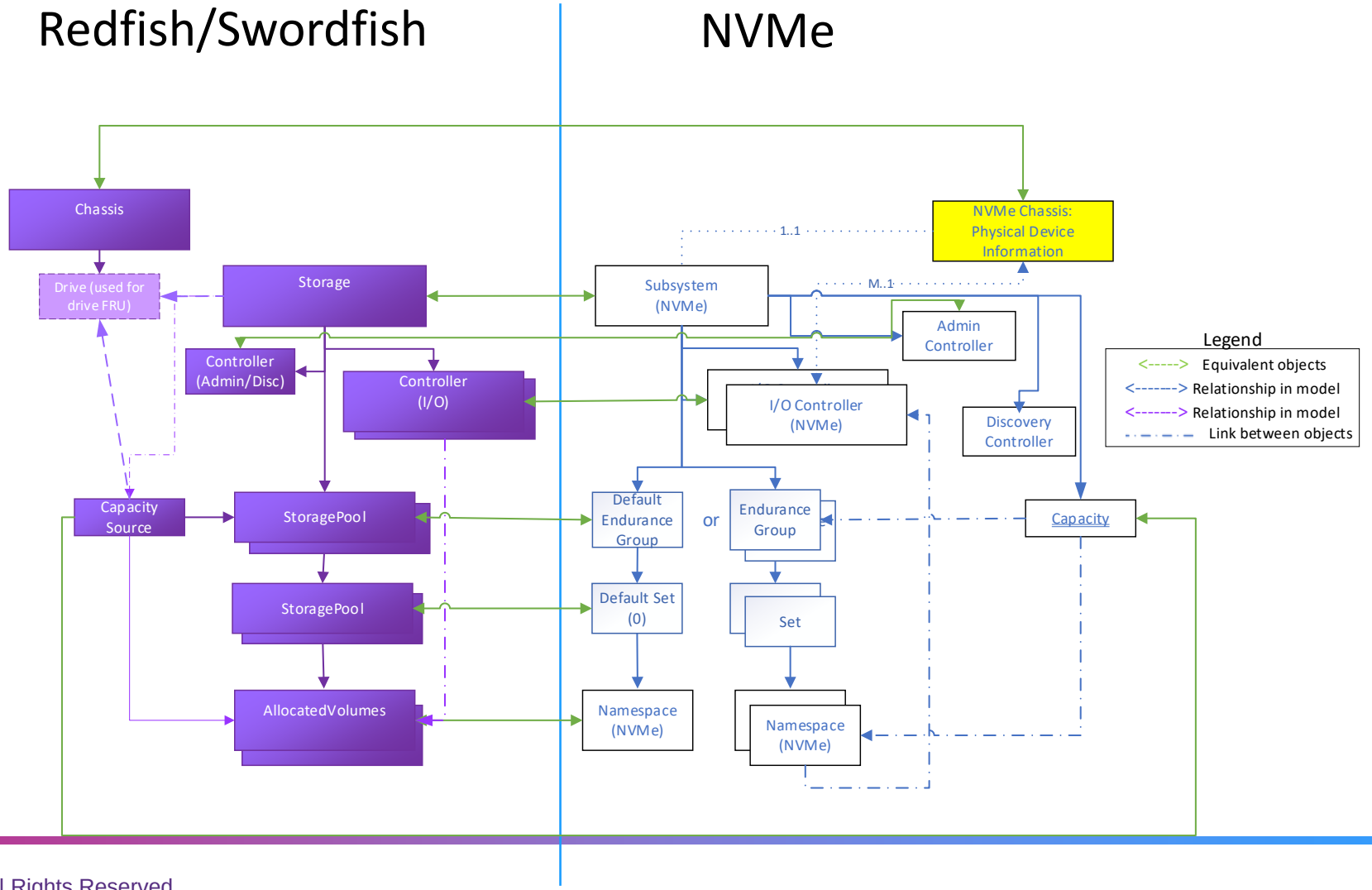
- The model diagrams do not reflect all available schema elements.

- Model leverages and coarsely maps to existing (Redfish and) Swordfish storage model

Major NVM Objects Mapped to RF/SF

- NVM Subsystem
  - An NVM subsystem includes one or more controllers, zero or more namespaces, and one or more ports. Examples of NVM subsystems include Enterprise and Client systems that utilize PCI Express based solid state drives and/or fabric connectivity.
- NVM Controller (IO, Admin and Discovery)
  - The interface between a host and an NVM subsystem
  - Admin controller: controller that exposes capabilities that allow a host to manage an NVM subsystem
  - Discovery: controller that exposes capabilities that allow a host to retrieve a Discovery Log Page
  - I/O: controller that implements I/O queues and is intended to be used to access a non-volatile memory storage medium
- Namespace
  - A quantity of non-volatile memory that may be formatted into logical blocks. When formatted, a namespace of size n is a collection of logical blocks with logical block addresses from 0 to (n-1)
- Endurance Group
  - A portion of NVM in the NVM subsystem whose endurance is managed as a group
- NVM Set
  - An NVM Set is a collection of NVM that is separate (logically and potentially physically) from NVM in other NVM Sets.
- NVM Domain
  - A domain is the smallest indivisible unit that shares state (e.g., power state, capacity information).
  - Domain members can be NVM controllers, endurance groups, sets or namespaces

NVMe Subsystem Model



Sample instantiations

Sample Instantiations – Mockups correspond to these

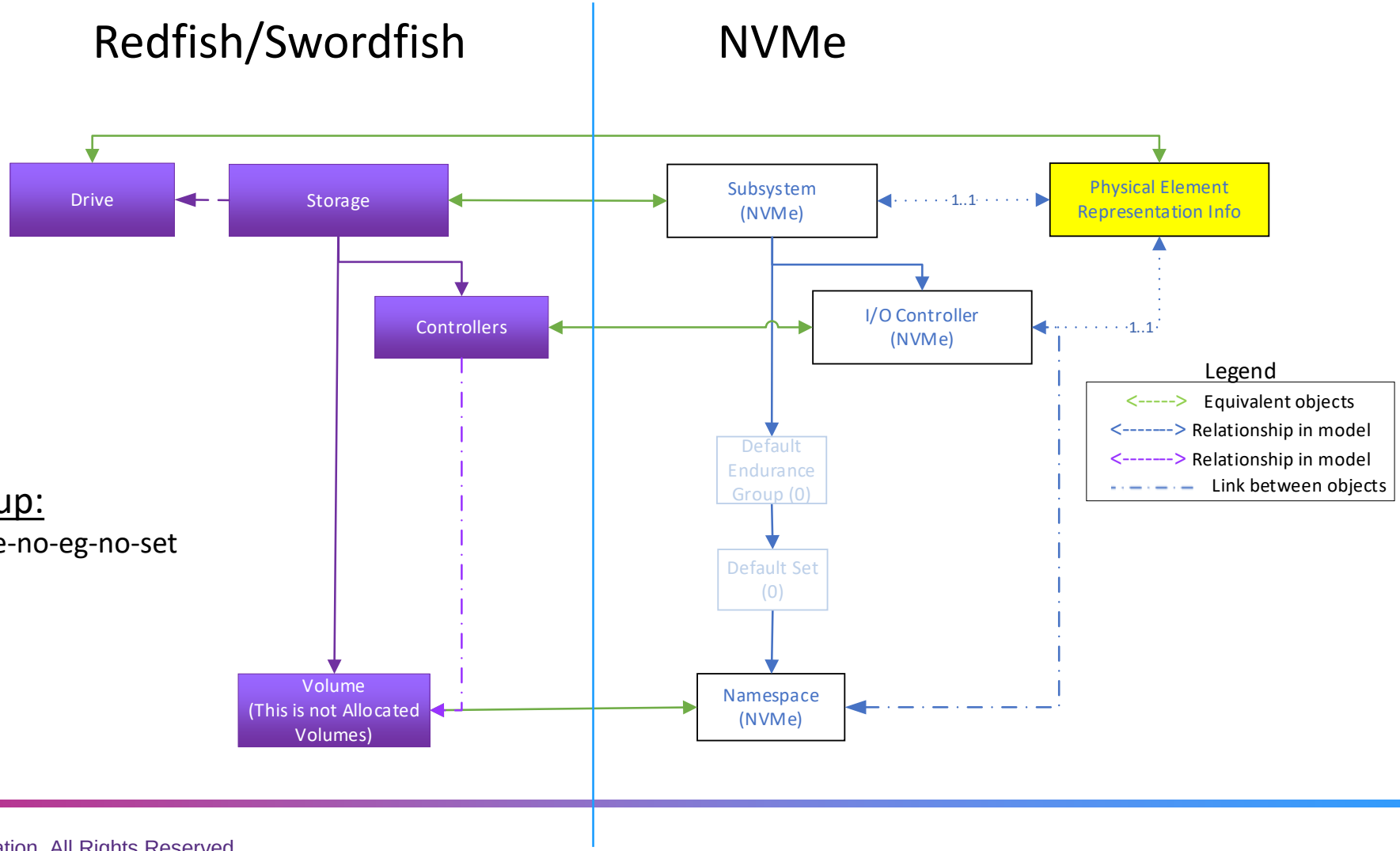
SNIA

Mapped Models and Documented Permutations

- Device Model – NVMe
  - Simple SSD
    - Default Endurance Group / Default Set
    - Single Endurance Group / Single Set
  - JBOF – PCIe front-end attach to set of drives
  - EBOF – Ethernet front-end attach to set of drives
  - Fabric Attach Array
    - RBOF – Simple RAID front-end attach to set of drives
  - Opaque Array
    - Front end is NVMe, back end is vendor choice (may incorporate existing technologies with NVMe)
- Subsystem (Fabric) Model – NVMe-oF
  - Fabric-attached subsystem presenting logical subsystem, controller, namespace, port and allowed host
  - Simple SSD with NVMe-oF Attach
    - Default Endurance Group / Default Set
  - NVMe Domains

Yellow – TBD Mockups

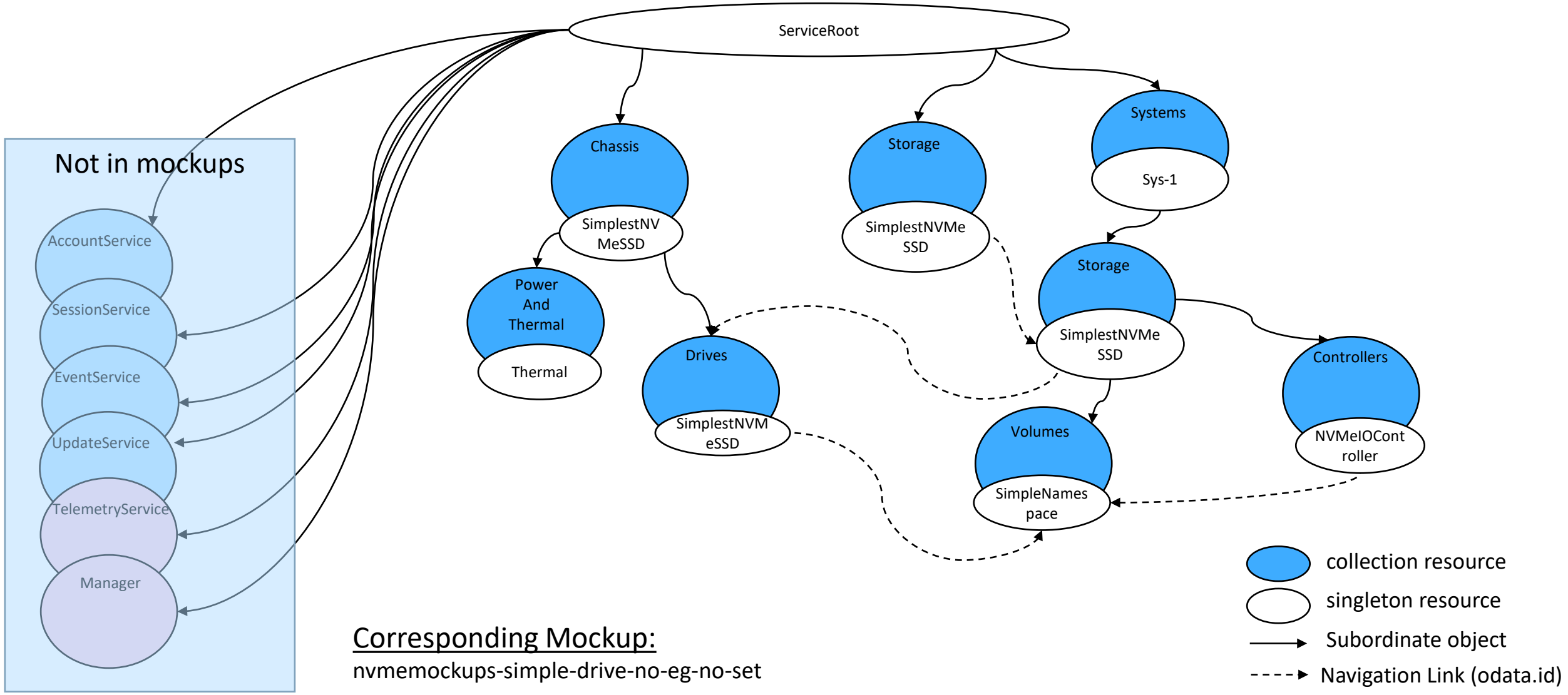
Simple SSD Implementation: Default Endurance Group and Set (Not Implemented in RF / SF)



<u>Corresponding Mockup:</u>

nvmemockups-simple-drive-no-eg-no-set

Simple NVMe Drive: No Namespace Mgmt, No EG, No Set



Corresponding Mockup:

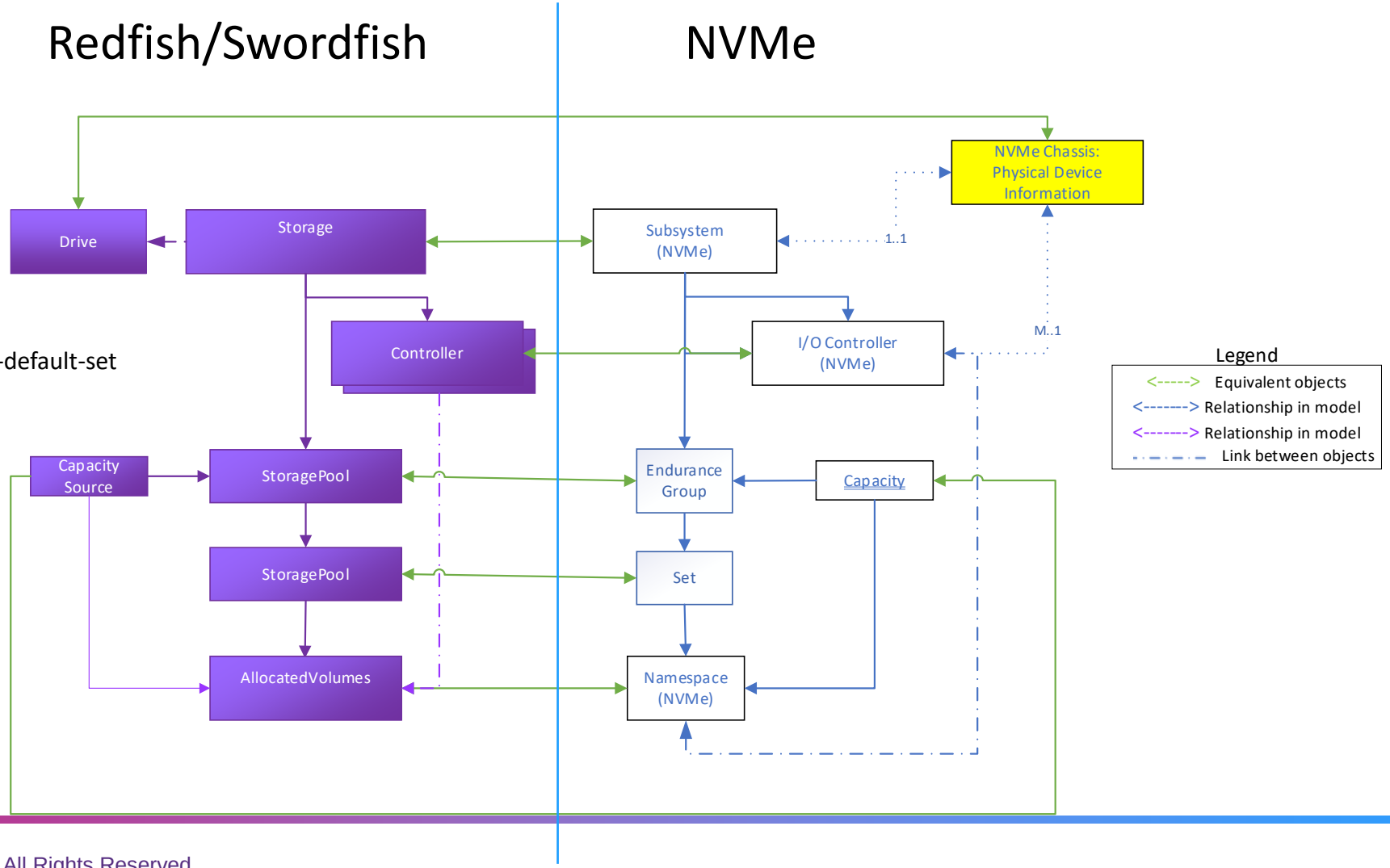
nvmemockups-simple-drive-no-eg-no-set

Management Interface for Devices Supporting Multiple Namespaces

- For devices that support multiple namespace creation:
  - If the device is not EG / Set aware (e.g., most that are NVMe version 1.3 or prior), creation of namespaces is done by POST to the Volume collection
    - (Q): This also applies to devices that only implement Default EG and Default Set behavior
  - For devices that are EG / Set aware and implement one or more EG and sets:
    - To create namespaces, POST to Set to create Namespace

Note: This is very similar to the guidance provided between RF and SF (POST to Volume Collection vs StoragePool)

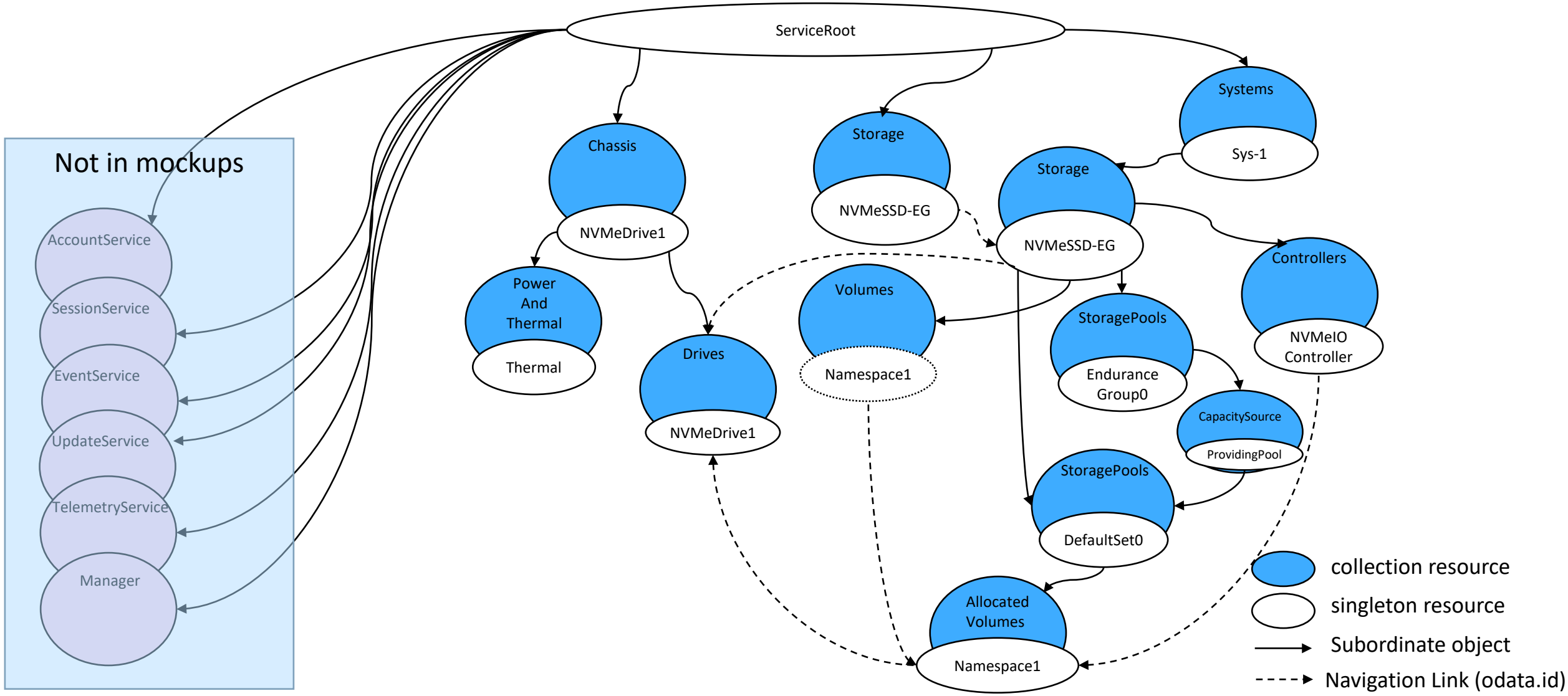
Simple SSD Implementation: Simple Endurance Group and Set



Corresponding Mockup:

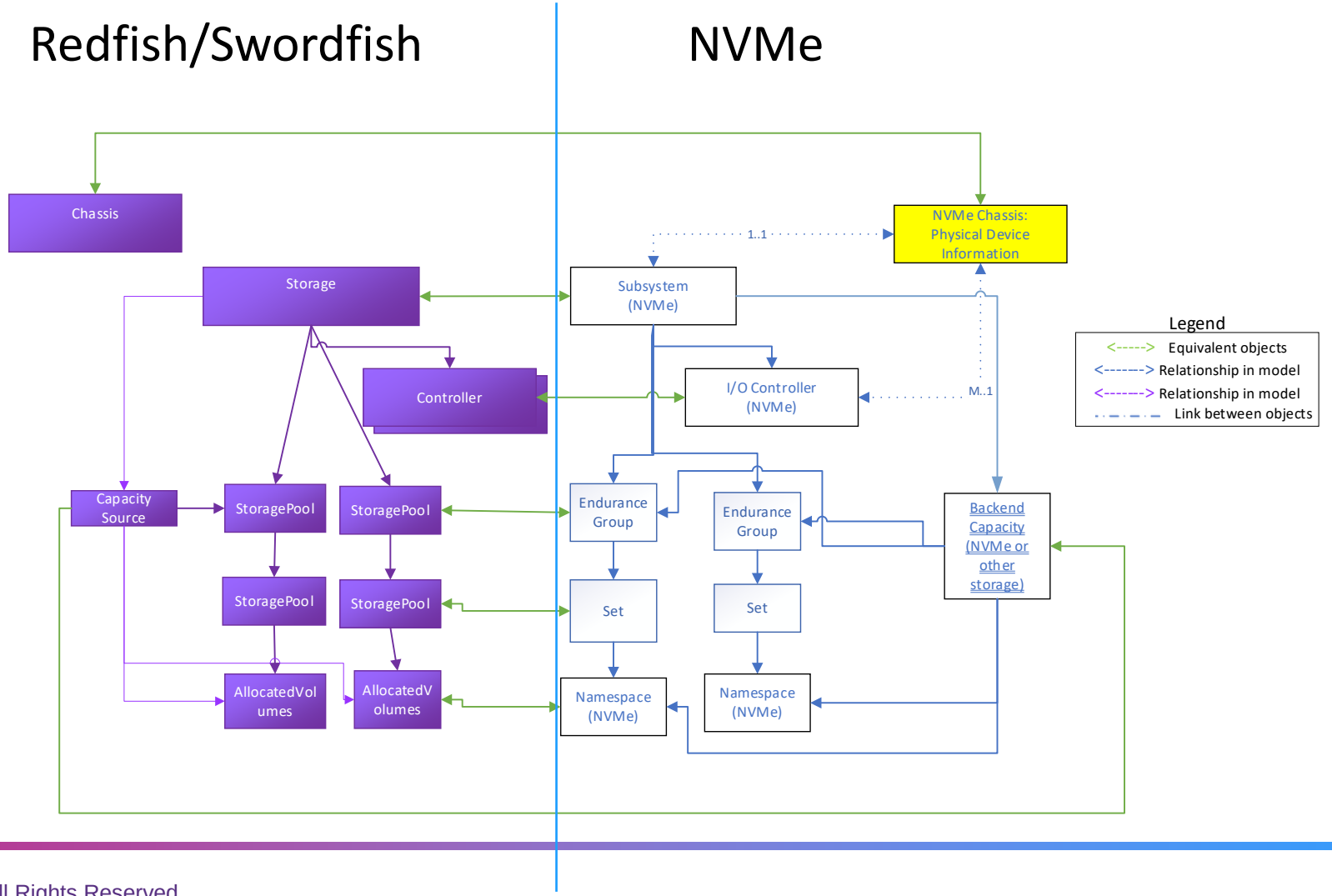
nvmemockups-simple-drive-default-eg-default-set

NVMe Drive: with EG and set

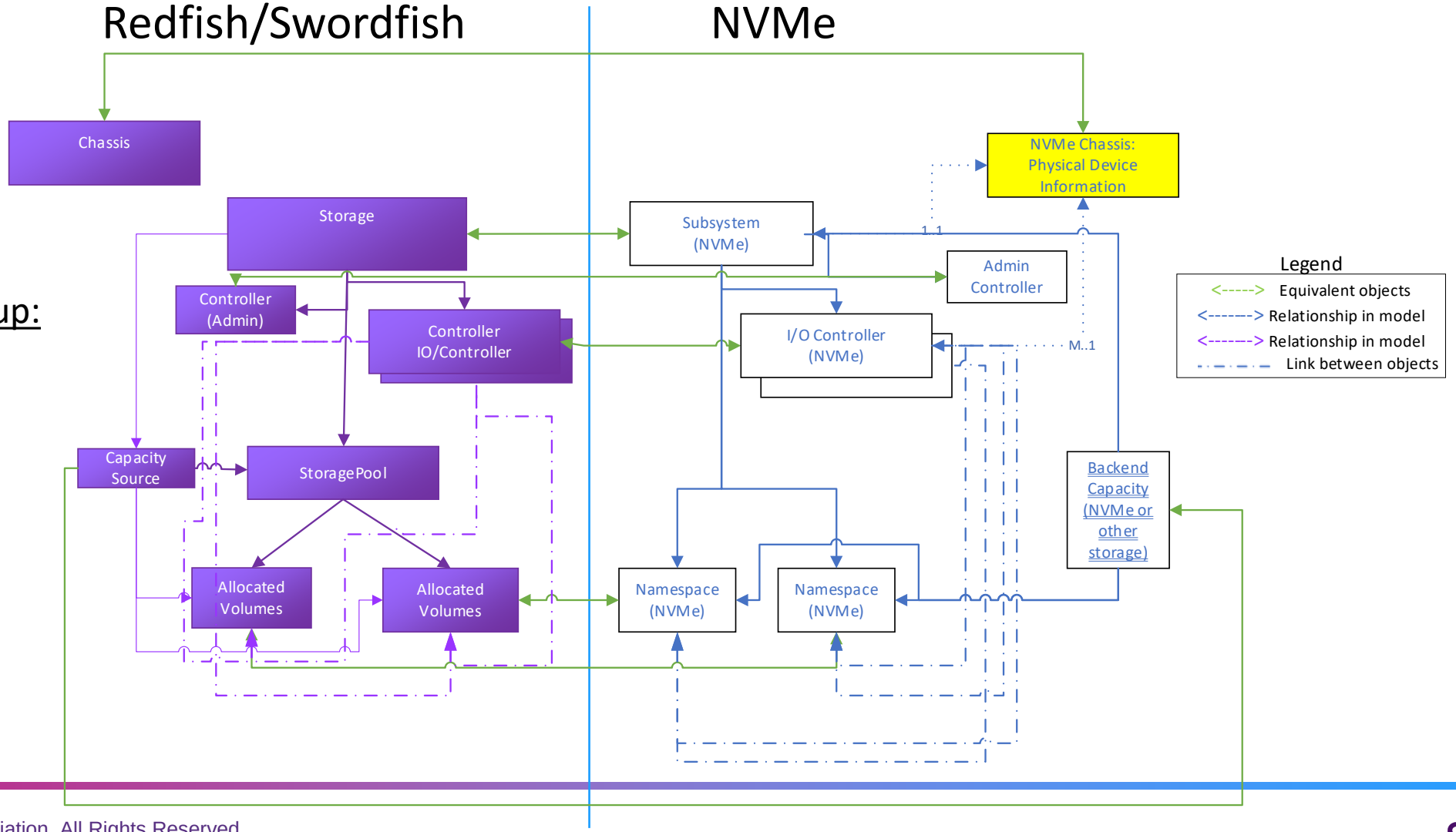


NVMe Array

<u>Corresponding Mockup:</u>
nvmemockups-fabric-attach-array



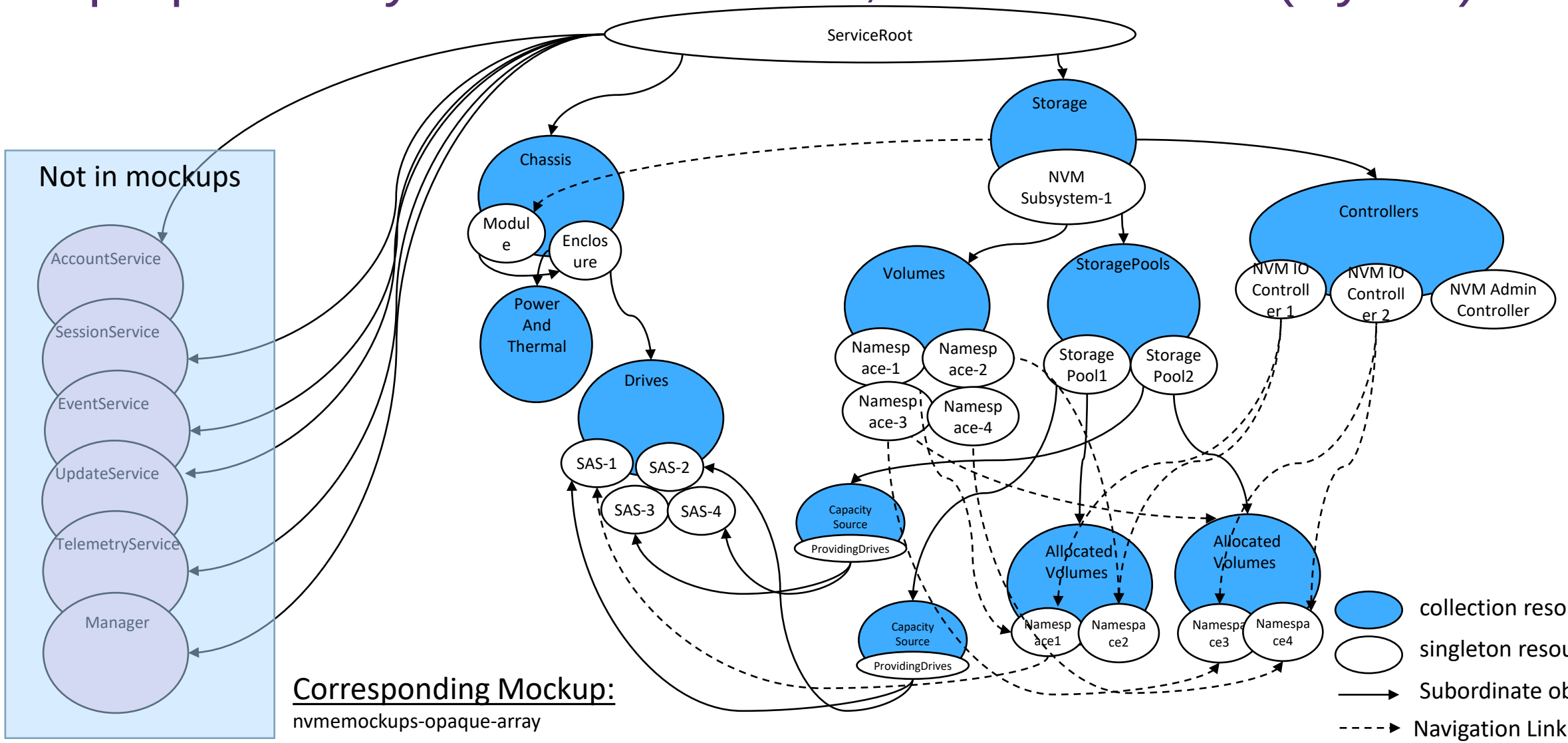
Opaque Array



Corresponding Mockup:

nvmemockups-opaque-array

“Opaque” Array: NVMe Front-end, SAS Backend (Hybrid)

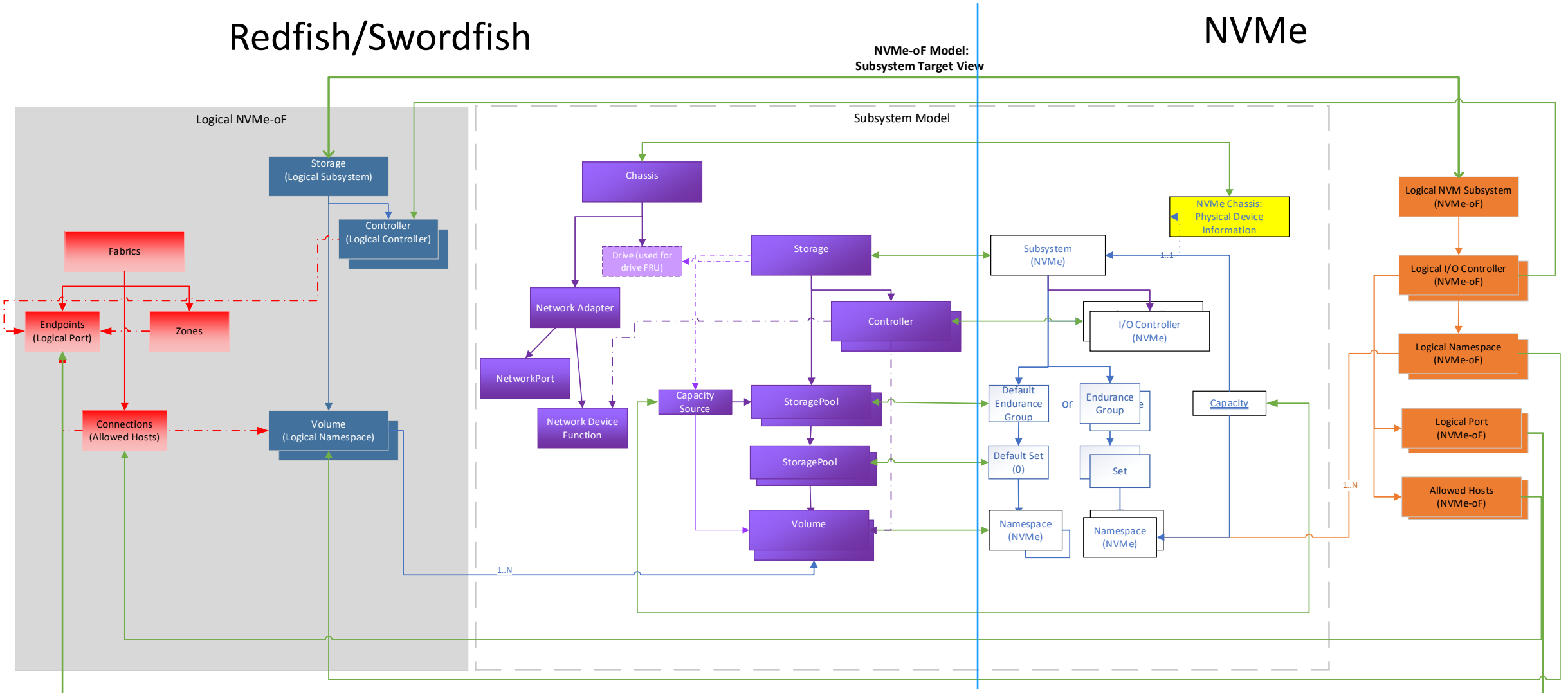


Corresponding Mockup:

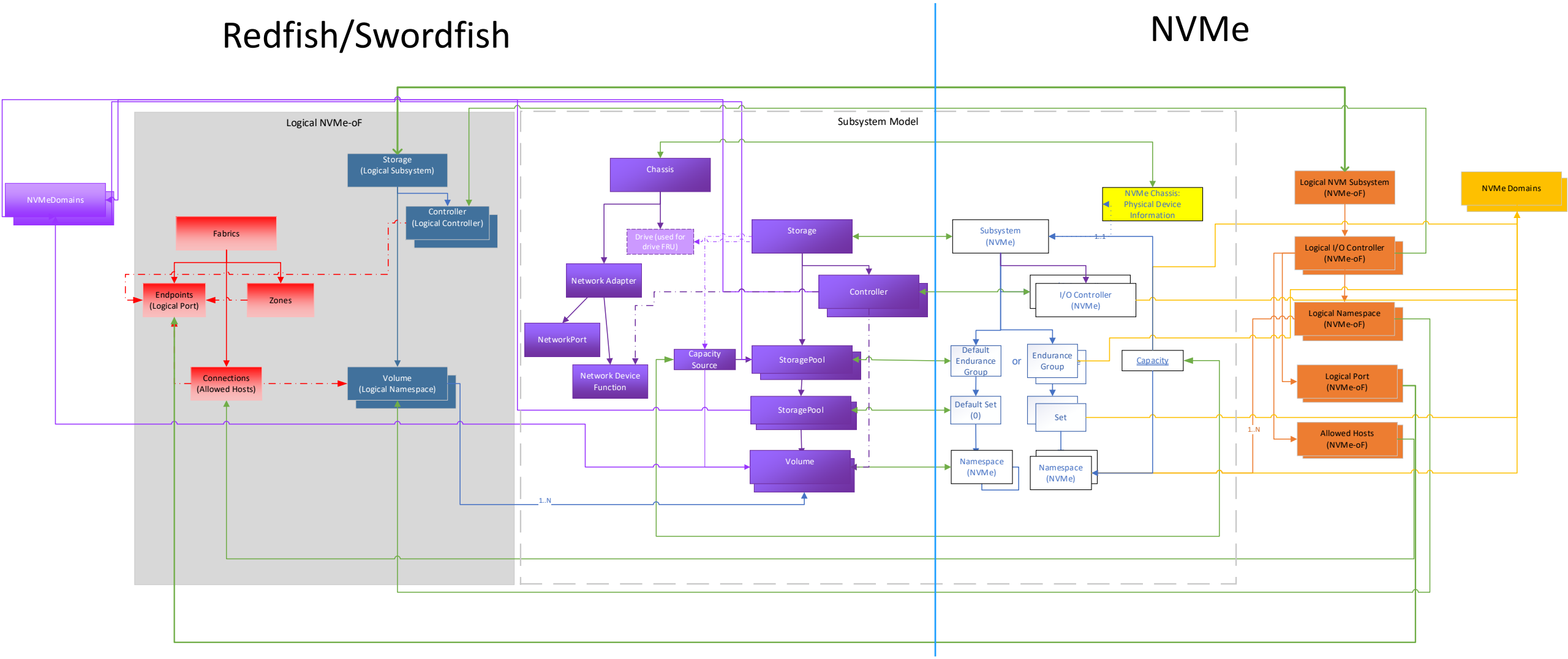
nvmemockups-opaque-array

NVMe-oF: Model

NVMe-oF: Subsystem Model



NVMe-oF: Model with Domains

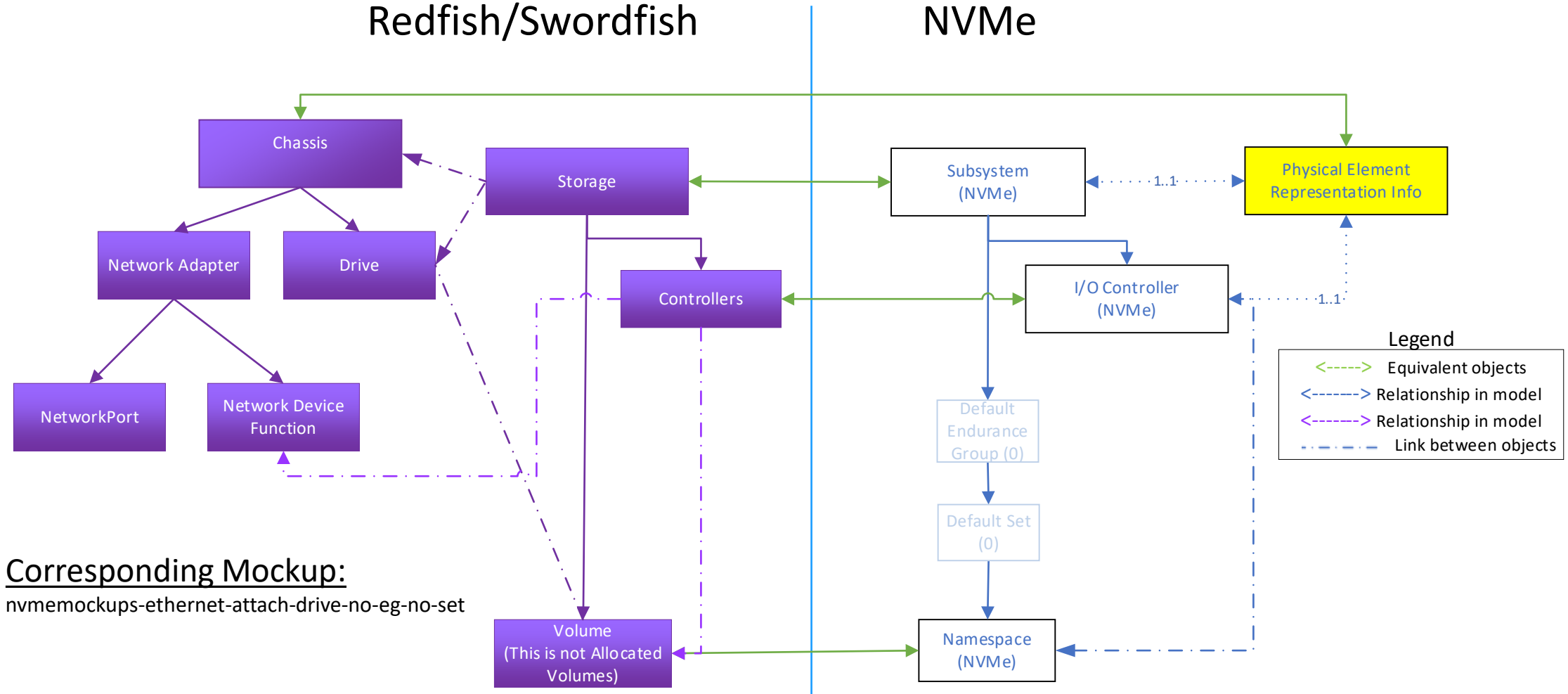


SNIA

Sample instantiations

Sample Instantiations – Mockups correspond to these

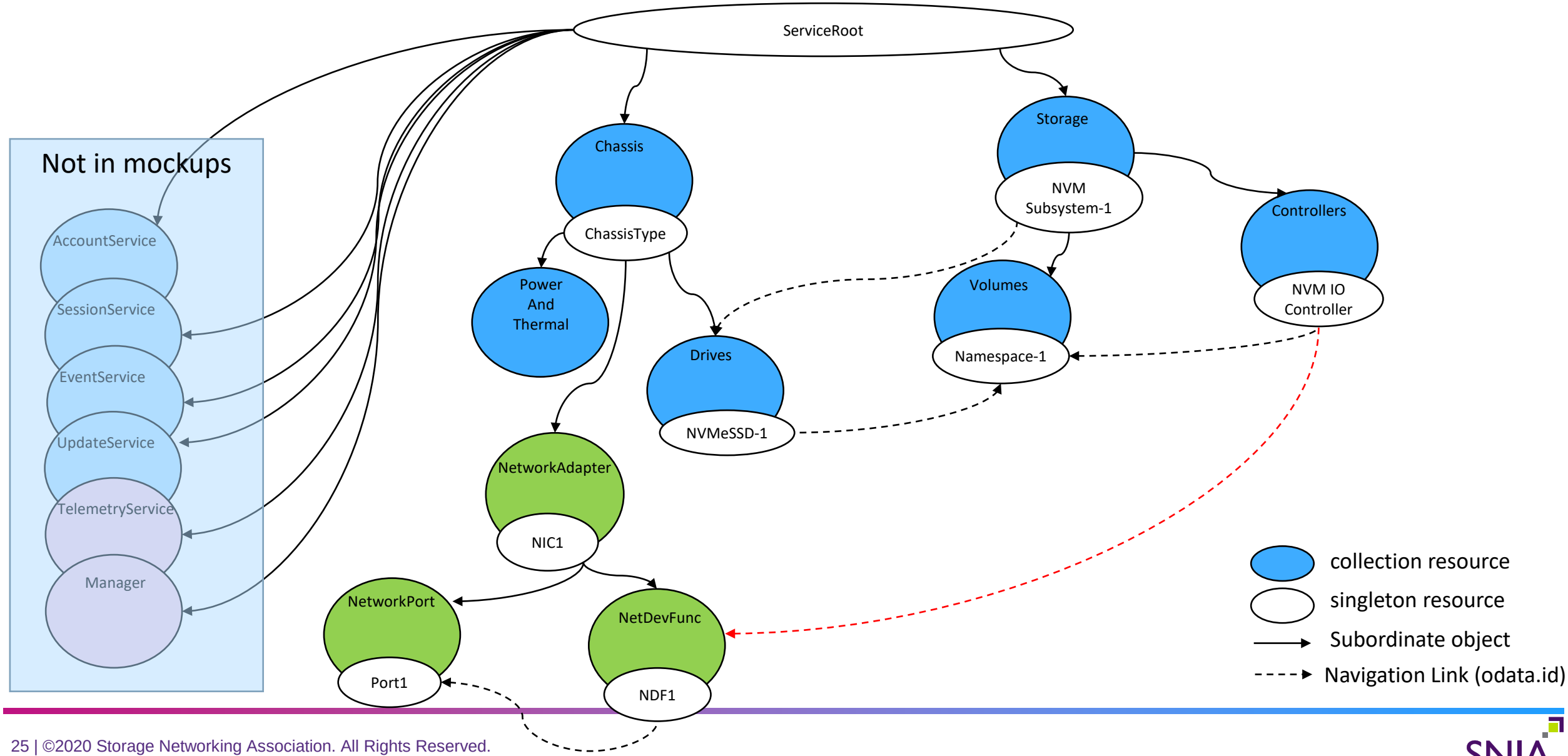
Simple SSD with NVMe-oF Attach (not NVMe TP 6011)



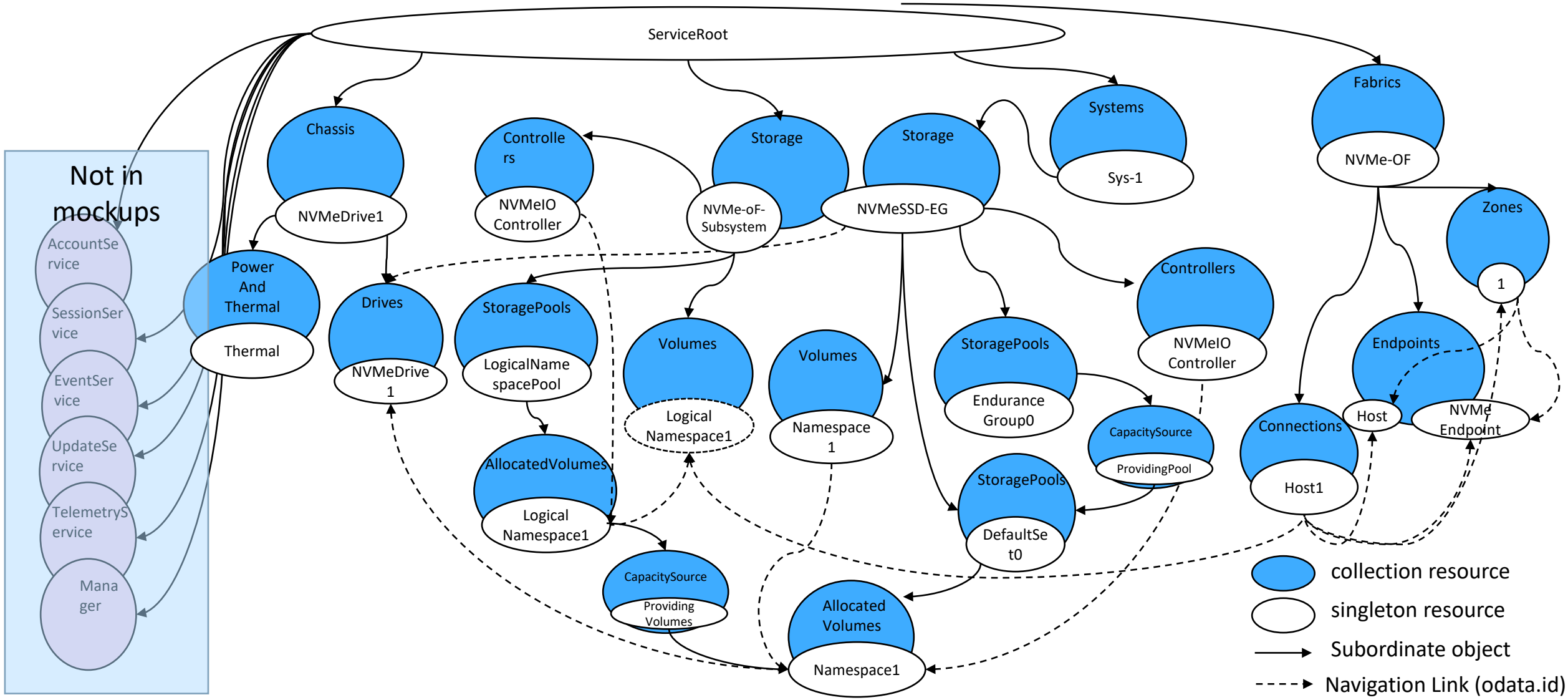
Corresponding Mockup:

nvmemockups-ethernet-attach-drive-no-eg-no-set

Simple NVMe Drive Network / Ethernet Attach (not TP 6011)



NVMe-oF: One Logical/Exported Subsystem from One Physical Subsystem



Additions to Schema

New Properties Added: Storage / NVMe Subsystem

- Storage (Subsystem)
  - Properties added to Storage for general use
    - Controllers [] – use instead of StorageController [] array
    - Identifiers – used for NQN / NGUID / etc. (unique ids)

New Properties / Schema Added for NVM Controllers

- StorageController (NVM Controller)
  - ControllerType: Admin/Discovery/IO
  - NVMeVersion
  - NVMeControllerAttributes
    - ReportsUUIDList
    - SupportsSQAssociations
    - ReportsNamespaceGranularity
    - SupportsTrafficBasedKeepAlive
    - SupportsPredictableLatencyMode
    - SupportsEnduranceGroups
    - SupportsNVMSets
    - SupportsExceedingPowerOfNonOperationalState
    - Supports128BitHostId
  - MaxQueueSize
  - ANACharacteristics
    - AccessState: Optimized / NonOptimized / Inaccessible / PersistentLoss
    - Volume
  - NVMeSMARTCriticalWarnings:
    - PMRUnreliable (Bool)
    - PowerBackupFailed (Bool)
    - MediaInReadOnly (Bool)
    - OverallSubsystemDegraded (Bool)
    - SpareCapacityWornOut (Bool)

NVMe Admin Controller

- Will be 0 or 1 admin controller
- Exists for in-band SES admin command processing
- What to show via RF/SF:
  - Use case: Part of comprehensive discovery / existence (extremely limited inventory info only)
  - May want to be able to reset an admin controller
- Can you create an admin controller if it doesn't exist?
  - No. It exists to provide access to fixed function services

NVMe Discovery Controller

- Is a subclass type of an NVMe controller
- May have 0, 1 or many controllers
- Created when hosts connects to the discovery NQN (via NVMe-oF “Connect” command)
  - Will be encouraging a move to persistent connections
  - These will then be dynamic discovery interfaces sending async events to the hosts
- Desired functionality through RF/SF:
  - Limited to inventory info
  - Ability to reset controller
- Actual discovery information is populated in fabric and connection/storage group

New Properties Added:
StoragePool Enhancements for Endurance Groups and Sets

- StoragePool (for Endurance Group)
  - PredictedMediaLifeLeftPercent
  - EndGrpLifetime
    - PercentUsed
    - EnduranceEstimate
    - DataUnitsRead
    - DataUnitsWritten
    - MediaUnitsWritten
    - HostReadCommandCount
    - HostWriteCommandCount
    - MediaAndDataIntegrityErrorCount
    - ErrorInformationLogEntryCount
- StoragePool (for Set)
  - SetIdentifier
  - OptimalWriteSizeBytes
  - EnduranceGroupIdentifier
  - Random4kReadTypicalNanoSeconds
  - UnallocatedNVMNamespaceCapacityBytes

New Properties Added: Volume for Namespace

- Volume (Namespace)
  - NamespaceId
  - IsShareable (Bool)
  - NamespaceFeatures
    - SupportsThinProvisioning
    - SupportsAtomicTransactionSize
    - SupportsDeallocatedOrUnwrittenLBError
    - SupportsNGUIDFieldReuse
    - SupportsIOPerformanceHints
  - NumberLBAFormats
  - FormattedLBASize
  - MetadataTransferredAtEndOfDataLBA
  - NVMeVersion

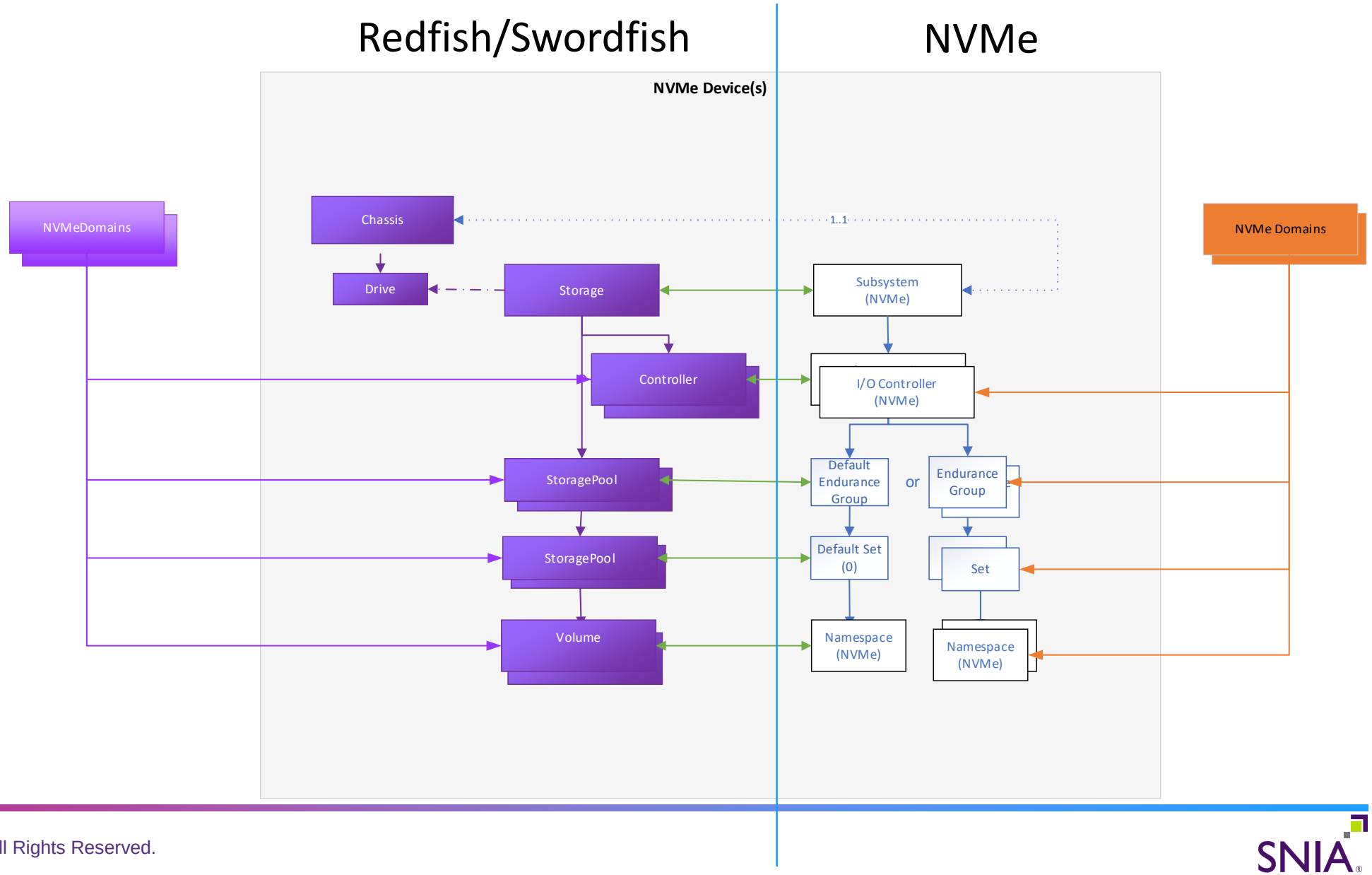
New Objects / Schema: NVMeDomain

- NVMeDomains to live off ServiceRoot
- Properties
  - DomainMembers
  - TotalDomainCapacityBytes
  - UnallocatedDomainCapacityBytes
  - MaximumCapacityPerEnduranceGroupBytes
  - AvailableFirmwareImages: []
    - Version, Vendor, NVMeDeviceType (sample properties, still TBD)
  - Links -> AssociatedDomains
- Behaviors defined in NVM Express TP 4009

NVMe Domains

Domain members can be NVM controllers, endurance groups, sets or namespaces

** Either physical or logical*



Future Items

Work in progress –potential properties to be considered for future releases

Managing IO Queues

- Controller Properties:

```
"AllocatedIoQueues": {
    "Submission": 64,
    "Completion": 64
},
```


Deferred Work Areas from v1

- LED
- Power profiles
- Defer mapping NVMe telemetry log page
  - Blob pass-through

- Redfish extension
  - NIC Teaming / Link aggregation in ACD
  - Network switch chassis aggregation (MLAG)

Additional Properties to Add (Align w/ RF)

- LED management
  - Comparison with SES LED mgmt / model
  - Review RF enhanced LED management proposal (Slawek)
  - Support for updated Redfish LED modeling (Locator)

Additional Properties to Add (Defer)

- Telemetry / log Data
  - Mapped to RF/SF schema / properties
    - SMART mapped properties
    - VPD Data (partial)
  - Binary blob pass-through
    - Vendor-unique (pure blob)
      - Mapped into OEM Extensions
    - Binary blob with NVMe standard defined header
    - VPD Data (partial)

ToDo: Look at standard NVMe telemetry definition to see if / how it can be mapped to RF/SF

Additional Properties to Add (Defer)

- Look at NVMe power states
  - RF Power is actual current power state: On/Off/PoweringOn/PoweringOff
- NVMe is selecting a profile for acceptable performance ranges supported – needs characterization as advanced power management functionality
  - Curtis: This is an internal set of properties, don't see a reason to expose via RF/SF at this point.
- Add NVM controller properties to power management model somewhere
  - RTD3ResumeLatencyuS
  - RTD3EntryLatencyuS
- Use Redfish Basic Power State info for devices
- ToDo: Add a new NVMe Power Profile equivalent model

