Intel® Rack Scale Design (Intel® RSD) Architecture Specification

Software v2.4

April 2019

Revision 001

Document Number: 608492-001

Contents

Figures

Tables

Revision History

Revision	Description	Date
001	Initial release for Intel® RSD Software release v2.4	April 2019

§

1.0 Introduction

This document contains information about the Intel® Rack Scale Design (Intel® RSD) software v2.4 reference architecture as a reference implementation using Intel Platform technology. This reference implementation demonstrates Intel's vision for an industry leading rack architecture that is modular and extensible, catering to the needs of the cloud, and other server market segments. Key objectives for Intel® RSD systems include a reduction in the Total Cost of Ownership (TCO), deliver agility at scale, and simplify data center operations.

1.1 Scope

This document provides the requirements, recommendations, and/or optional criteria for Intel® RSD systems. Intel® RSD Application Program Interface (API) conformance is described in the Intel® RSD Pooled System Management referenced in Table 3.

Note: This document contains the NVMe* over Fabric (NVMe*-oF) related contents on top of the Intel® RSD .v2.4 Architecture Specification.

1.2 Intended Audience

The intended audience for this specification includes designers and engineers working with the Intel® RSD Software .v2.4 release, such as:

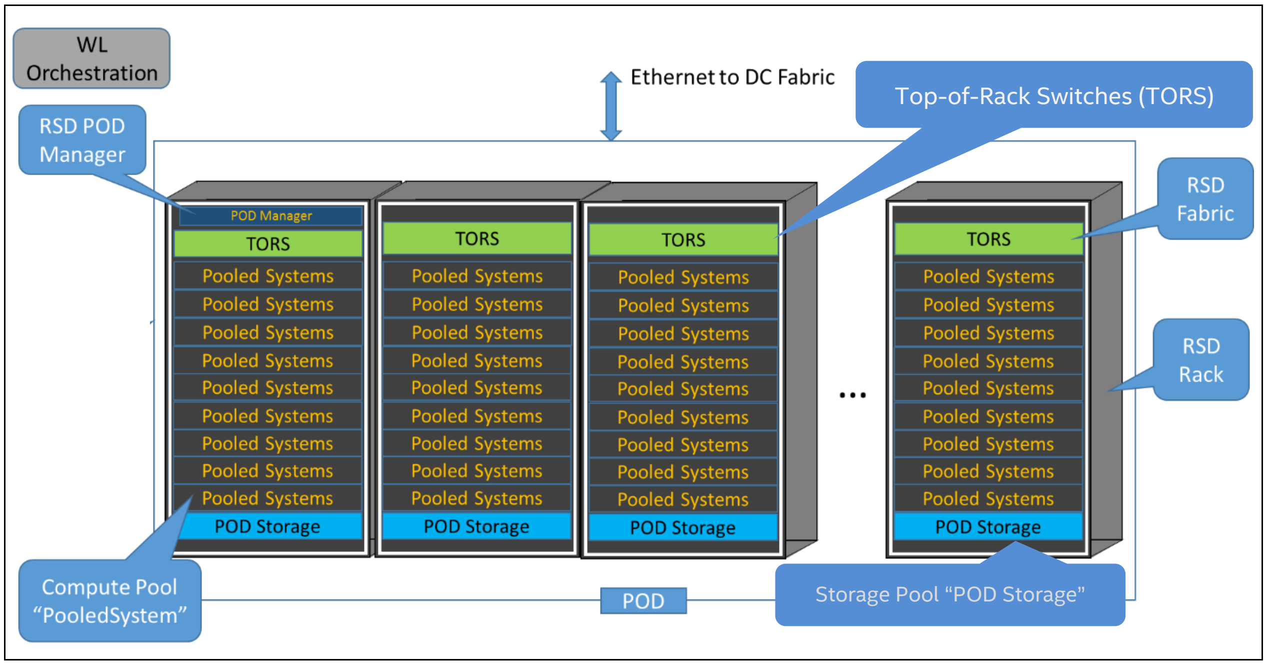
- Hardware vendors (Original Equipment Manufacturers (OEMs)/ Original Design Manufacturers (ODMs)) who will build Intel® RSD platforms or Intel® RSD Platform components that are integrated into the Intel® RSD Platform.
- Software vendors (for example, Independent Software Vendors (ISVs)/ Independent Basic Input/output System (BIOS) Vendors (Independent BIOS Vendors (IBVs)) who will implement the Intel® RSD API on Intel® RSD Platform hardware components and on supporting components.

1.3 Intel® RSD Platform Overview

Figure 1 illustrates the various elements of the Intel® RSD Platform. In this scenario, the POD consists of several racks with resources that allow for POD scaling. These resources (compute resources labeled as `"Pooled Systems"` and storage resources labeled as `"POD Storage"`) are contained in multiple Racks. At the top of each Rack, there is an interconnecting fabric (labeled as Top-Of-Rack Switch (TORS)) that provides the Intel® RSD fabric connection among the racks. One Rack (on the left in this diagram) contains the PODM software that conducts the workload orchestration (WL Orchestration) and coordinates the activity in the POD.

Note: The PODM must not shut down on rack-wide resets or power downs. This can be achieved by installing separate power to the PODM with highly available power such as a battery backup, redundant power, or PODMs set up in a redundant configuration.

Figure 1. Intel® RSD POD Block Diagram



1.3.1 Major Changes in Intel® RSD v2.1

The major changes added from the Intel® RSD v1.2 to the Intel® RSD v2.1 definition are:

- Peripheral Component Interconnect express* (PCIe*) Direct Attach Pooled Input/output (I/O)
- Software components requirements

1.3.2 Major Changes in Intel® RSD v2.2

The major changes added from Intel® RSD v2.1 to the Intel® RSD v2.2 definition are:

- Trusted Platform Module* (TPM*) discovery
- Field-programmable gate array (FPGA) discovery
- Deep discovery of the platform components and capabilities
- Network Management Services
- In-band BIOS/firmware update blocking
- Telemetry

1.3.3 Major Changes in Intel® RSD v2.3

The major changes added from Intel® RSD v2.2 to the Intel® RSD v2.3 definition are:

- The addition of the Ethernet Pooled Storage (NVMe over Fabrics using Remote Direct Memory Access (RDMA))
- The deprecation of Deep discovery feature. The feature may be removed in future releases.

1.3.4 Major Changes in Intel® RSD .v2.3.1

- Added clarifications to the backward compatibility information
- Added clarifications to the security sections

1.3.5 Major Changes in Intel® RSD v2.4

- Added Pooled FPGA over PCIe*
- Added Pooled FPGA over Ethernet
- The Deep Discovery feature was removed

1.3.6 Architecture Terminology Definitions

Table 1 provides definitions for the architecture terminology used in this document.

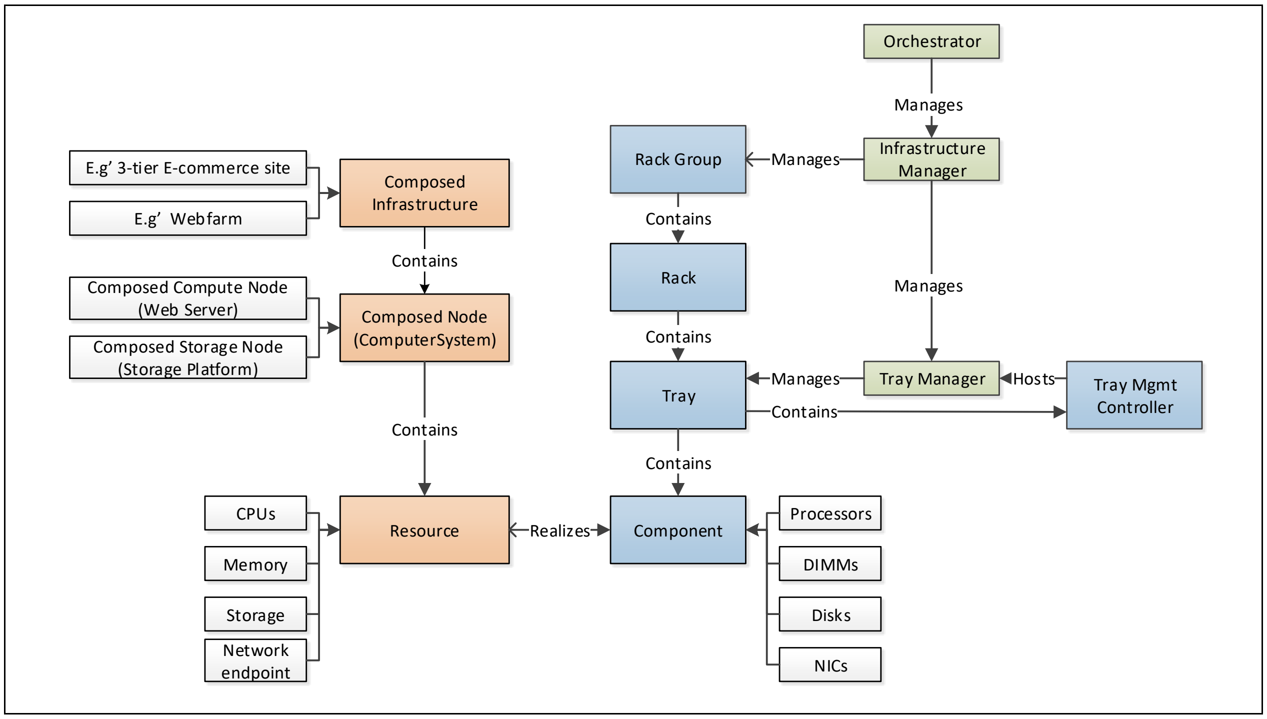
Table 1. Intel® RSD Architecture Terminology Definitions

Term	Definition
Blade	A physical element in a Drawer that is a grouping of compute or storage resources. **NOTE:** The terms Module and Blade are used interchangeably because the physical elements can overlap in degenerate cases.
Composed Node	A logical collection of physical system resources, composed by the PODM. The PODM creates Composed Nodes within the rack by communicating with the PSME to request the allocation of resources based on the user input.
Compute Blade	A type of Blade server that provides compute resources within a Compute Drawer.
Compute Drawer	A type of Drawer that holds compute resources for a POD.
Compute/Storage Module	A physical compilation of compute or storage hardware resources that represent a field replaceable unit (FRU) or that can be reset together.
Drawer	A physical element in a Rack that acts as a container for Modules, Blades, or other system resources. Drawers provide the underlying compute or storage resources which create Composed Nodes.
In-band vs. out-of-band control	A protocol control system. In-band designs pass control data on the same connection as the main data (examples of In-band control include Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP)). Out-of-band designs separate control data from the main data (an example of Out-of-band control is File Transfer Protocol (FTP)).
Module	A module is a physical element in a Drawer that is a grouping of compute or storage hardware resources that represent a FRU or that can be reset together. **NOTE:** The terms Module and Blade are used interchangeably because the physical elements can overlap in degenerate cases.
Node	A logical system element of the Intel® RSD architecture that contains resources (CPU components, memory components, or switches) that attach to a computer or a network and can perform basic RSD functions.
PNC	Pooled Node Controller. A physical system element providing connectivity and access control between the CPUs and the pool of NVMe* storage, FPGAs, and Accelerators.
POD	A physical and/or virtual collection of Racks, Drawers, Modules, and Blades.
Rack	A Rack is a physical element in a POD that holds POD resources.
RMM	Rack Management Module
Storage Drawer	A type of Drawer that holds storage resources for a POD.
Network Agent	A logical agent that manages the Intel® RSD-compliant switch.
Storage Blade	A type of Blade server that provides additional storage resources within a Storage Drawer.
TPM	Trusted Platform Module: An international standard for a secure microprocessor that is dedicated to hardware security.
Platform	A physical element of computer system architecture that includes a microprocessor, instruction set architecture design, microarchitecture design, logic design, and implementation that allows users to develop, run, and manage applications without adding the complexity of building and maintaining an infrastructure. Systems designed around the x86-microprocessor architecture are considered a Platform.
Rack Manager	A logical system element (firmware) that runs on the Rack Management Controller (RMC).

1.3.7 Logical Representation of Intel® RSD POD

Figure 2 illustrates a logical representation of the Intel® RSD POD.

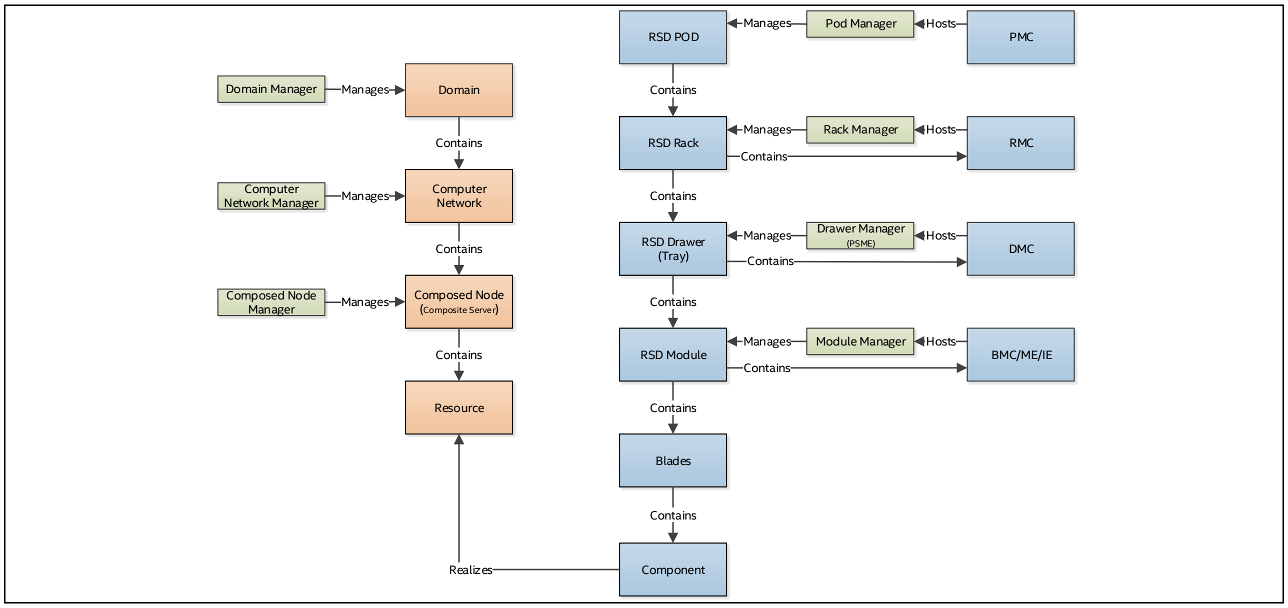
Figure 2. Logical View of the Intel® RSD POD



1.3.8 Management Elements of Intel® RSD POD

Figure 3 illustrates the management elements of an Intel® RSD POD.

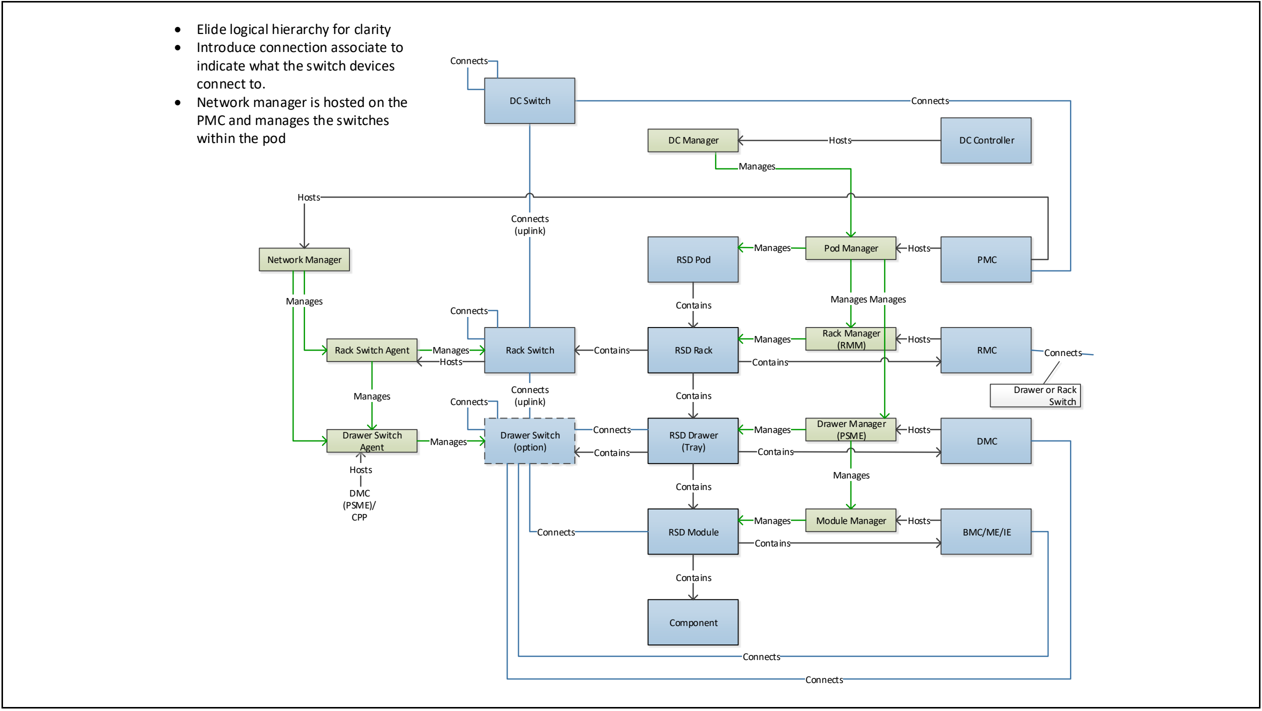
Figure 3. Management Elements of a Logical Intel® RSD POD



1.3.9 North Bound Management Hierarchy of Intel® RSD POD

Figure 4 illustrates the North Bound (NB) management hierarchy of an Intel® RSD POD.

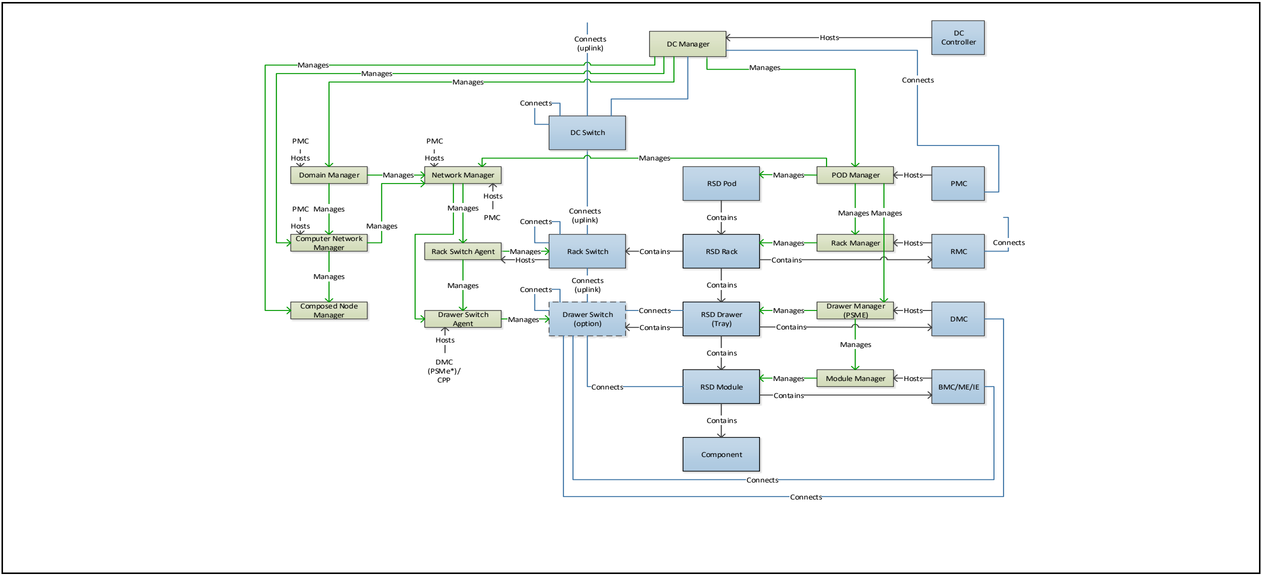
Figure 4. North Bound Management Hierarchy of Intel® RSD POD



1.3.10 Full Management Hierarchy of Intel® RSD POD manager

Figure 5 illustrates the full management hierarchy of an Intel® RSD POD.

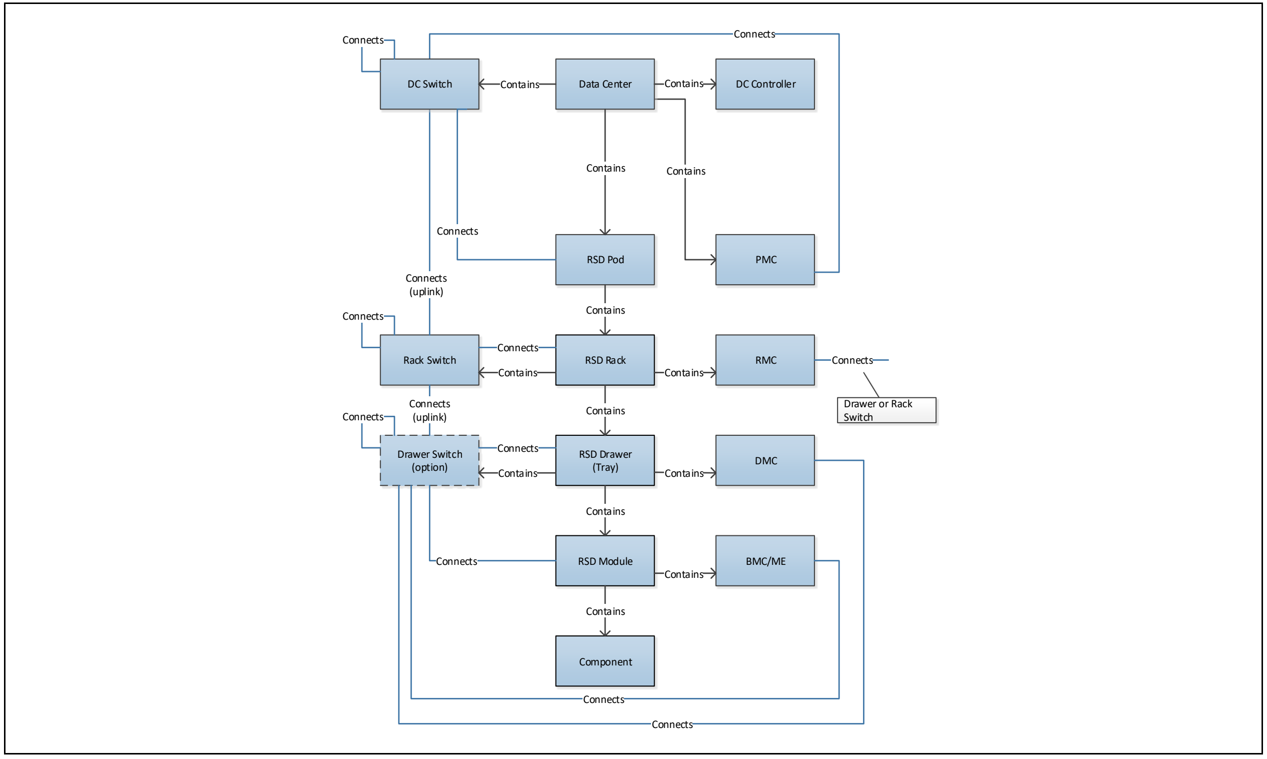
Figure 5. Full Management Hierarchy of Intel® RSD POD



1.3.11 Full Physical Containment Hierarchy of Intel® RSD POD

Figure 6 illustrates the full Physical Containment Hierarchy of an Intel® RSD POD.

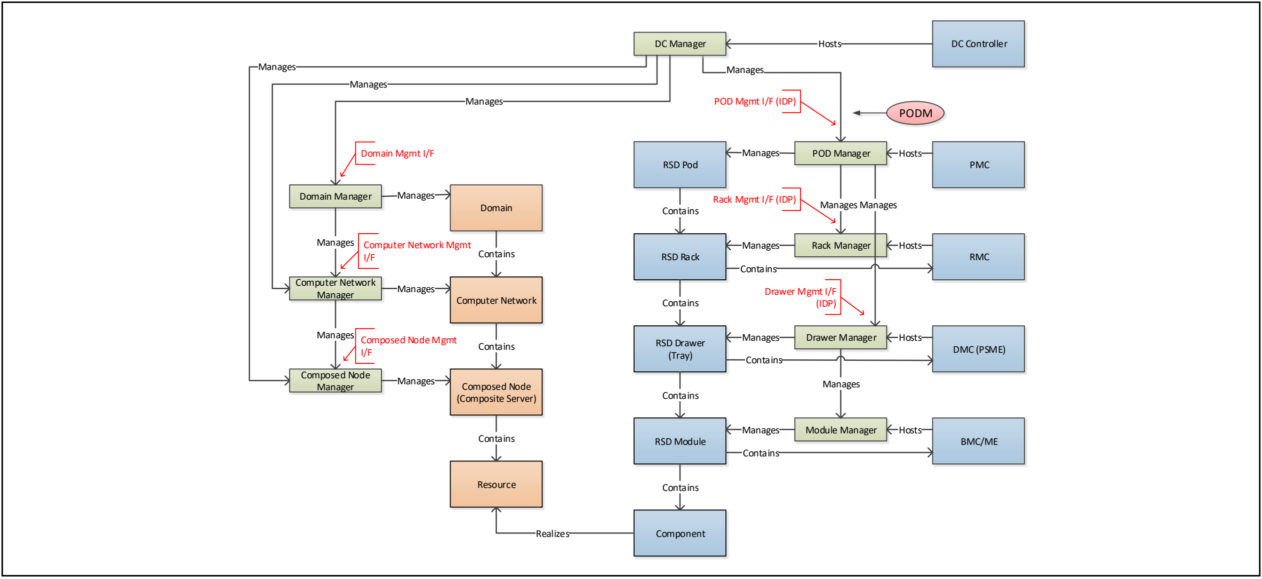
Figure 6. Full Physical Containment Hierarchy of Intel® RSD POD



1.3.12 Interfaces of Intel® RSD POD

Figure 7 illustrates the interfaces of an Intel® RSD POD.

Figure 7. Interfaces of Intel® RSD POD



1.3.13 Mapping POD Software to POD Architecture Layers

Figure 8 illustrates the five Physical Layers for a typical POD built on the Intel® RSD Architecture, and how the software and firmware span different levels of this Architecture.

The PSME is responsible for Drawer identification management, as well as supporting the Intel® RSD PSME API and communicating with the Baseboard Management Controller (BMC) to perform Compute/Storage Module-level management.

1.3.14 Intel® RSD Container Options

Even though Figure 3 through Figure 9 illustrate Intel® RSD Architecture with container hierarchy, including:

- an RSD POD containing an RSD Rack
- an RSD Rack containing an RSD Drawer
- an RSD Drawer containing an RSD Module
- an RSD Module containing an RSD Blade

The RSD Drawer, RSD Module, and RSD Blade are optional components. For example, if an Intel® RSD Module implements the PSME API, the RSD PODM can interface with the RSD Module without having an RSD Drawer.

Figure 8. Physical Layers and Software Architecture in Intel® RSD with Physical Rack/RMM

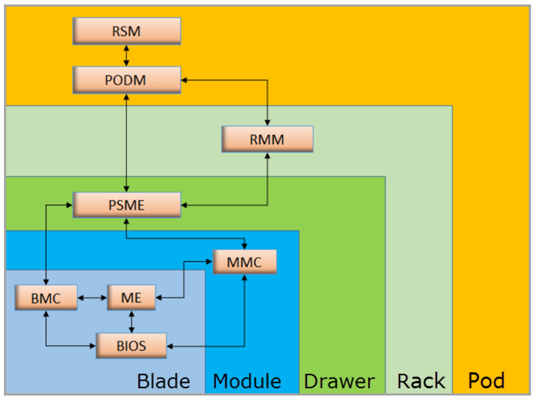
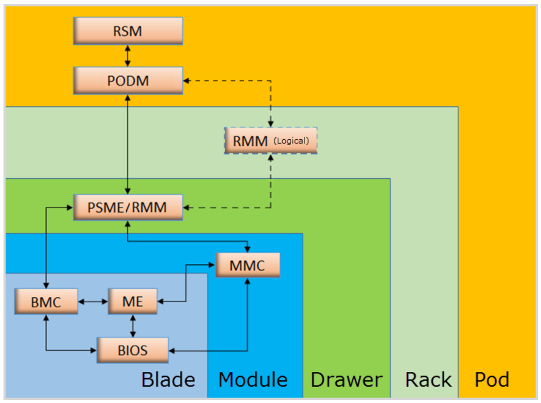


Figure 9 illustrates an alternative system configuration that does not include the separate instance of the RMM in the Rack. In this case, one of the PSME instances in the POD can be used to provide the RMM services.

Figure 9. Physical Layers and Software Architecture in Intel® RSD with Logical Rack/RMM



1.4 Intel® RSD Platform Hardware Components

The Intel® RSD Platform consists of the following Hardware Components:

RSD POD

Where the PODM resides:

- RSD Rack (logical or physical)
- RSD Drawer
- RSD Module/Blade
- PSME
- RSD Compute/Storage Module:

  A Compute Module or a Storage Module:

  - Power supply (either independent per Drawer or shared between Drawers)
  - Cooling (either independent per Drawer or shared between Drawers, using fans or other means)
  - Pooled Node Controller (PNC)

1.5 Intel® RSD Platform Software Components

The Intel® RSD Platform consists of the following Software Components:

- PODM
- RMM Software
- PSME Software (optional if BMC provides Redfish* interface for Compute Nodes)
- Network Switch agent Software
- BMC Firmware
- BIOS
- PNC Firmware

1.6 Intel® RSD API

The Intel® RSD API information is outlined in the Intel® RSD API specifications listed in Table 3.

1.7 Conventions

The key words/phrases "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in *Key Words for Use in RFCs to Indicate Requirement Levels*, March 1997, RFC2119, Table 3.

1.8 Notes and Symbol Convention

Symbol and note convention are similar to typographical conventions used in the *Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol An Interface for Managing Cloud Infrastructure specification*, Table 3.

Notation used in JSON* serialization description:

Note: Mandatory in italics indicate data types instead of literal Mandatory.

- Characters are appended to items to indicate cardinality:
  - "?" (0 or 1)

  - "*" (0 or more)
  - "+" (1 or more)
- Vertical bars, "|", denote choice. For example, "a|b" means a choice between "a" and "b".
- Parentheses, "(" and ")", are used to indicate the scope of the operators "?", "*", "+" and "|".
- Ellipses (i.e., "...") indicate points of extensibility.

Note: The lack of ellipses does not mean no extensibility point exists; rather it is not explicitly called out.

1.9 Acronym Definitions

Table 2 contains definitions for the acronyms used in this document.

Table 2. Intel® RSD Acronym Definitions

Term	Definition
ACM	Authenticated Code Module. A security feature that is used to establish trust-level before executing the code. The processor or hardware first measures a digitally signed module (the ACM) that was provided by the chipset manufacturer. The ACM then measures the first BIOS module, and the first BIOS module then measures the next module, and so on.
AFU	Accelerator Function Unit. An FPGA that uses the AFU bit stream to provide end user functionality.
API	Application program interface. A set of routines, protocols, and tools for building software applications. API defines operations, inputs, and outputs.
ASIC	Application Specific Integrated Circuit
BIOS	Basic Input/output System. Firmware that initializes and tests Compute/Storage Module hardware components, and loads a bootloader or an operating system from a mass memory device. The BIOS supports the UEFI interface.
BMC	Baseboard Management Controller. A specialized service processor that monitors the physical state of a computer and provides services to monitor and control certain Compute/Storage Module operations. The BMC supports the Intelligent Platform Management Interface (IPMI) or Redfish.
BW	Bandwidth
CIMI	Cloud Infrastructure Management Interface
CMDB	configuration management database
DC	Drawer Controller
DHCP	Distributed Host Configuration Protocol. A standard network protocol used to dynamically distribute network configuration parameters, for instance IP addresses.
DNS	Domain Name System
DWPD	Drive writers per day
EFI	Extensible Firmware Interface
FPGA	A field-programmable gate array (FPGA) is an integrated circuit designed to be configured after manufacturing.
FPGA-oF*	FPGA-over Fabric
FRU	Field Replicable Unit
FTP	File Transfer Protocol
FW	Firmware
GUID	Globally Unique Identifier
HA	High Availability. RSD performance requirements may dictate the use of High Availability (HA) designs as a redundancy feature. If a Rack supports HA RMM, then (by definition) there is more than one RMM present in the Rack. If the primary RMM fails, then the secondary RMM takes over immediately.
HBA	Host Bus Adapter
HDD	Hard Disk Drive
HTTP	Hypertext Transfer Protocol
HTTPS	Hyper Text Transfer Protocol with Secure Sockets Layer

Term	Definition
IA	Intel® Architecture
IB	In-band
IBV	Independent BIOS Vendor
IE	Innovation Engine. The physical controller in the Module or Blade.
IPMI	Intelligent Platform Management Interface
I/O	Input/output
IOPS	Input/output Per Second
Intel® ME	Intel® Management Engine. A physical hardware resource that gives access to hardware features at the baseboard level below the OS.
Intel® RSD	Intel® Rack Scale Design
Intel® TXT	Intel® Trusted eXecution Technology.
iPXE	An open-source implementation of the PXE client firmware and bootloader.
ISV	Independent software vendor
iSCSI	Internet Small Computer Systems Interface
JBOD	Just a Bunch Of Disks. A collection of storage devices consolidated in one chassis for easy serviceability.
KVM	keyboard, video, and mouse
NB	North Bound. Used to identify a direction of information transfer between the top and bottom layers of the software. From the PSME point of view, a transfer from the PSME to the PODM is considered a NB transfer.
NIC	Network Interface Card
NTP	Network Time Protocol. An Internet protocol used to synchronize the clocks of computers to a common time reference.
NVDIMM	Non-Volatile Dual In-line Memory Module
NVMe*-oF	NVMe* over Fabric. The NVMe drives are attached through network fabric such as Ethernet.
NVMe*-MI	NVMe* Management Interface. An out of band management interface to manage NVMe.
MCTP	Management Component Transport Protocol
MM	Module Manager. Firmware that runs on BMC/Intel ME/IE
MMC	Module Management Controller. The controller that manages the Blades in the module.
ODM	Original Design Manufacturer
OEM	Original Equipment Manufacturer
OOB	Out-of-Band
PCIe*	Peripheral Component Interconnect express*
PDoS	Permanent Denial of Service
PFMS	Pooled FPGA Management Service
PM	Persistent Memory. Byte-accessible persistent memory that is produced by either DDR-based NVDIMM or high-density NVM technology.
PNC	Pooled Node Controller. A physical system element that provides connectivity and access control between the CPUs and the pool of NVMe storage, FPGA, and Accelerators.
POD	A logical and/or physical collection of Racks within a shared infrastructure management domain.
PODM	POD Manager. The software that manages logical groupings of functionality across all infrastructure in a POD.
PSMS	Pooled Storage Management Service. An agent running on the pooled system provides the PSME interface for manageability.
PSME	Pooled System Management Engine. System management software that runs on the DMC and is responsible for the configuration of pooled Storage Modules by the PNC, the network (SDN), the Compute Modules, and the switches.
PXE	Preboot eXecution Environment. A specification that allows devices to boot over a network.
QoS	Quality of Service
RAID	Redundant Array of Inexpensive/Independent Disks
RAS	Reliability, Availability, and Serviceability
REST	REpresentational State Transfer

Term	Definition
RDMA	Remote Direct Memory Access
RMC	Rack Management Controller. A physical system element that provides Rack management features.
RMM	Rack Management Module. A physical system element that is responsible for managing the Rack, which normally assigns IDs for the instances of PSME in the Rack, and manages the Rack power and cooling.
Intel® RSD	Intel® Rack Scale Design
RSD API	Refers to PODM REST API, RMM REST API, PSME REST API combinations.
RTM	Root of Trust for Measurement. A security feature; a component that can be trusted to reliably measure and report to the Root of Trust.
SATA*	Serial ATA*
SB	South Bound. Used to identify a direction of information transfer between the top and bottom layers of the software. From the PODM point of view, a transfer from the PODM to the PSME is considered a SB transfer.
SDN	Software-Defined Network
SMBIOS	System Management BIOS
SMTP	Simple Mail Transfer Protocol
SOL	Serial-Over-LAN. A mechanism that allows a serial port's input/output to be redirected over IP.
SPDK	Storage Performance Development Kit
SRIS	Separate Refclk Independent SSC. A reference clock forwarding system.
SSD	Solid State Drive
SSDP	Simple Service Discovery Protocol
SSH	Secure SHell. An encrypted network protocol (versions: SSH-1, and SSH-2) that can be used to secure a remote login and other network services.
SW	Software
TCO	Total Cost of Ownership
TCP	Thermal Design Power
TLS	Transport Layer Security. A security protocol that provides connection security.
TORS	Top-of-Rack Switch. A physical switch in each rack that connects the racks together to handle NB/SB traffic flow.
TPM*	Trusted Platform Module*.
UEFI	Unified Extensible Firmware Interface

1.10 References to More Information

To find more information about related aspects of Intel® RSD. Refer to the documents listed in Table 3.

Note: The following documents are placeholders for final documents.

Table 3. Reference Documents and Resources

Doc ID	Title	Location
608486	*Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) User Guide Software v2.4*	Note: *https://www.intel.com/content/www/us/en/architecture-and-technology/rack-scale-design/rack-scale-design-resources.html*
608487	*Intel® Rack Scale Design (Intel® RSD) Conformance and Software Reference Kit Getting Started Guide v2.4*	
608488	*Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) Release Notes Software v2.4*	
608489	*Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) User Guide Software v2.4*	
608490	*Intel® Rack Scale Design (Intel® RSD) Pooled System Management (PSME) Release Notes Software v2.4*	
608491	Intel® Rack Scale Design Storage Services API Specification Software v2.4	

Doc ID	Title	Location
608493	±Intel® Rack Scale Design (Intel® RSD) Pod Manager (PODM) Representational State Transfer (REST) API Specification Software v2.4	
608494	± Intel® Rack Scale Design (Intel® RSD) Rack Management Module (RMM) Representational State Transfer (REST) API Specification Software v2.4	
608495	*Intel® Rack Scale Design (Intel® RSD) Generic Assets Management Interface (GAMI) API Specification v2.4*	
608496	±*Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) REST API Specification Software v2.4*	
608497	*Intel® Rack Scale Design (Intel® RSD) Conformance Test Suite (CTS) Release Notes*	
608298	*Field Programmable Gate Array (FPGA) over Fabric Protocol Architecture Specification*	https://cdrdv2.intel.com/v1/dl/getContent/608298
596167	Intel® Rack Scale Design (Intel® RSD) for Cascade Lake Platform Firmware Extension Specification	https://cdrdv2.intel.com/v1/dl/getContent/596167
332200	*Intel® Intelligent Power Node Manager 4.0 External Interface Specification Using IPMI*	*http://www.intel.com/content/www/us/en/power-management/intelligent-power-node-manager-3-0-specification.html*
N/A	*Hypertext Transfer Protocol - HTTP/1.1*	https://dmtf.org/sites/default/files/standards/documents/DSP0266_1.1.0.pdf
N/A	*NVM Express over Fabrics Specification*	https://nvmexpress.org/wp-content/uploads/NVMe-over-Fabrics-1_0a-2018.07.23-Ratified.pdf

Note:

1. Documents referenced in this table which have a Document ID, but cannot be accessed, can be obtained by calling 1-800-548-4725 or by visiting *www.intel.com/design/literature.htm* obtain a copy..
2. ±In the text, Intel® RSD Specifications refer to the following documents:
   - Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) Representational State Transfer (REST) API Specification Software v2.4.
   - Intel® Rack Scale Design (Intel® RSD) Rack Management Module (RMM) Representational State Transfer (REST) API Specification Software .v2.4.
   - Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) Representational State Transfer (REST) API Specification Software .v2.4.

§

2.0 Intel® RSD Platform Requirements Summary

This section provides a summary of the Intel® RSD Platform design requirements - refer to Table 4. The columns contain links to specific topics in this document.

Note: The Intel® RSD version column contains the initial version of the Intel® RSD when the specified feature is introduced.

Table 4. Intel® RSD Platform Architecture Requirements Summary

Section	Intel® RSD Validation Criteria	Compliance	Type	Intel® RSD version
3.0	Intel® RSD Platform General Guidelines			
3.2	Generic Intel® RSD Platform Requirements			
3.2.1	Rack Must Have One or More Logical Pooled System Management Engine (PSME) Software	Required	Software	1.2
3.2.2	Shared or Highly Efficient Power Supply	Required	Hardware	1.2
3.2.3	Shared or Highly Efficient Cooling	Recommended	Hardware	1.2
3.2.4	Just a Bunch of Disks (JBOD) Support	Optional	Hardware	1.2
3.2.5	Compute Module with Local Boot Drive	Optional	Hardware	1.2
3.2.6	At Least One Intel® RSD Compute Module in POD	Required	Hardware	1.2
3.2.7	Compute Module Serviceability Independence	Required	Hardware	1.2
3.2.8	Ethernet-Based Fabric for Management and External Network Connectivity	Required	Hardware	1.2
3.2.9	At Least One Ethernet Switch in the POD	Required	Hardware	1.2
3.2.10	Network Switch Support For Network Software Agent	Required	Hardware	1.2
3.2.11	PODM Support PNC Capabilities	Recommended	Hardware	2.1
3.2.12	Platform Support For FPGAs	Recommended	Hardware	2.2
3.2.13	Hot-Pluggable Modules in Intel® RSD Drawers	Required	Hardware	1.2
3.2.14	Backward-Compatibility for Intel® RSD v2.3 PODM	Required	Software	2.1
3.2.15	Backward-Compatibility for Intel® RSD v2.3 Drawer	Required	Software	2.1
3.2.16	Intel® RSD Versions Coexistence Support within a Rack	Required	Software	2.1
3.2.17	PODM-to-PSME Communication Channel Protection	Required	Software	1.2
3.2.18	PODM-to-RMM Communication Channel Protection	Required	Hardware	1.2
3.2.19	PSME-to-RMM Communication Channel Protection	Required	Hardware	1.2
3.2.20	In-Band Reconfiguration of the Private Network	Required	Hardware	2.2
3.2.21	User-Maintained Backup Copy of Data	Recommended	Software	2.1
3.3	Intel® RSD Components Location Identification Support			
3.3.1	Field Replaceable Units Identification and Location Information	Required	Software	1.2
3.3.2	Connectivity Identification	Required	Software	2.1
3.4	Intel® RSD Fabric and Network Configuration			
3.4.1	OOB Management Network and In-Band Data Network Separation	Required	Hardware	1.2
3.4.2	Secure NTP Access Availability	Required	Software	2.1
3.4.3	Secure DHCP Server Availability If DHCP Discovery Is Used	Recommended	Software	2.1
3.4.4	Secure DNS Support	Recommended	Software	2.1
3.5	Intel® RSD Platform Configuration and Provisioning			
3.5.1	Serial Over LAN (SOL) or KVM Support for Compute Modules	Required	Software	2.1
3.5.2	Firmware and Software Updates Signed and Checked	Required	Software	2.2
3.5.3	PODM or Admin Control of Updates	Recommended	Software	2.2
3.6	Intel® RSD Platform Security			

Section	Intel® RSD Validation Criteria	Compliance	Type	Intel® RSD version
3.6.2	Composed Node Volatile Memory Clearing	Required	Software	2.1
3.6.3	User to Archive Data Before Decomposing a Node	Recommended	Software	2.1
3.7	Intel® RSD Power and Cooling			
3.7.1	Power Monitoring Support	Required	Software	2.1
3.7.2	Power Budgeting Support	Recommended	Software	1.2
3.7.3	Cooling Failure Reporting	Required	Software	2.2
4.0	Intel® RSD API			
4.1	Intel® RSD API Interface			
4.1.1	Intel® RSD API Compliance	Required	Software	1.2
4.1.2	Intel® RSD API Support for Access Control and Secure Communication Channel	Required	Software	2.1
5.0	Module Design Guidelines			
5.1	Module Reset, Power, and Performance			
5.1.1	Module Power On/Off Support	Required	Hardware	2.1
5.1.2	Module Reset Support	Required	Hardware	2.1
5.1.3	Power Monitoring Support	Required	Hardware	2.1
5.1.4	Power Budgeting Support	Recommended	Software	1.2
5.2	Module Features			
5.2.1	Expose TPM Capabilities if TPM Present	Required	Software	2.2
5.2.2	Expose SMBIOS Information to PSME	Required	Software	2.2
5.2.3	Expose FPGA Capabilities if FPGA Is Present	Required	Software	2.2
5.2.4	BIOS/Firmware Support for PNC if PNC Supported	Required	Software	2.1
5.2.5	Minimum 10 GbE Network Interface Card (NIC) per Module for Data Plane	Recommended	Hardware	2.1
5.3	Module Firmware Update			
5.3.1	Module In-Band Firmware Update Blocking from Composed System User	Required	Software	2.2
5.3.2	Firmware Update Authentication	Required	Software	2.2
5.3.3	Module Configuration Default Support	Recommended	Software	2.2
5.4	Module Configuration Information			
5.4.1	Module Processor Information (Out-of-Band)	Required	Software	2.2
5.4.2	Module Memory Information (Out-of-Band)	Required	Software	2.2
5.4.3	Module Storage Information (Out-of-Band)	Recommended	Software	2.2
5.4.4	Compute Module Remote OS Boot Support	Required	Software	2.1
5.4.5	Compute Module iPXE Support	Recommended	Software	2.1
5.4.6	Compute Module iSCSI Support	Recommended	Software	2.1
5.4.7	Compute Module OS Boot from Local Storage	Recommended	Software	2.1
5.4.8	Module Security Support Information	Recommended	Software	2.2
5.5	Reliability Availability and Serviceability (RAS) Support			
5.5.1	Out-of-Band Health Event Support	Required	Software	2.2
5.5.2	Error Persistence over Reboot	Recommended	Software	2.2
5.5.3	Reporting Health and Performance Information	Required	Software	2.2
6.0	PCIe* Direct Attach Pooled I/O Design Guidelines			
6.2	System Topology and Mapping			
6.2.1	Enumeration of Components in the System Must Be Deterministic and Persistent Across Power or Initialization Cycles	Required	Software	2.1
6.2.2	PSME Exclusive Management Link to PNC	Required	Hardware	2.1
6.2.3	Expose and Enumerate PNC Devices in a Pooled System	Required	Software	2.1

Section	Intel® RSD Validation Criteria	Compliance	Type	Intel® RSD version
6.2.4	Expose PSME Mapping of Management Connections to PNCs	Required	Software	2.1
6.2.5	Assignment of Primary PSME for PNC	Required	Software	2.1
6.2.6	Expose and Enumerate PNC Upstream Ports	Required	Software	2.1
6.2.7	Expose and Enumerate PNC Downstream Ports	Required	Software	2.1
6.2.8	Expose Data Path Cross-connections between Multiple PNCs	Recommended	Software	2.1
6.2.9	Expose and Enumerate Device Slots of the I/O Pooled system	Required	Software	2.1
6.2.10	Expose Mapping of Device Slot Connectivity to PNC Downstream Ports	Required	Software	2.1
6.2.11	Compute Module to PNC Upstream Port Connection ID Mapping	Required	Software	2.1
6.2.12	Expose the Connection Presence of Each Upstream Port	Optional	Software	2.1
6.3	I/O Device Discovery Support			
6.3.1	Expose the Presence of an I/O Device	Required	Software	2.1
6.3.2	Discovery of Device Type and Capability	Required	Software	2.1
6.3.3	PSME Configuration of I/O Device Support if Sharing of I/O Device Supported	Required	Software	2.1
6.3.4	Expose Metrics for PCIe Devices	Recommended	Software	2.3
6.4	I/O Device Assignment to Compute Module			
6.4.1	Full Assignment of a Device PCIe* Function to a Single Compute Node	Required	Software	2.1
6.4.2	Assignment of Single PCIe* Function to Multiple Upstream Ports	Optional	Software	2.1
6.4.3	Dynamic Assignment of a Device Shall Not Affect Other Device Connectivity	Required	Software	2.1
6.4.4	Dynamic Release of a Device Shall Not Affect Other Device Connectivity	Required	Software	2.1
6.4.5	Devices with Data Storage Must Secure Data Upon Release	Recommended	Software	2.1
6.4.6	User Data Must Be Purged Upon Release of a Programmable Device	Required	Software	2.4
6.4.7	Functional Blocks programmed by a Compute Module Must be Purged Upon Release of a Programmable Device	Required	Software	2.4
6.4.8	Persistence of Functional Blocks Instances programmed by the PSME for Programmable Devices is under the control of the PSME	Required	Software	2.4
6.4.9	I/O Resources Must Be in an Unassigned State Prior to Assignment to a Compute Node	Required	Software	2.1
6.5	Adding or Removing Devices from the I/O Pool			
6.5.1	Physical Hot Add Support of Devices to the I/O Pool	Required	Software	2.1
6.5.2	Managed Removal of Device from the I/O Pool Support	Required	Software	2.1
6.5.3	Surprise Removal of a Device from the I/O Pool Support	Required	Software	2.1
6.5.4	Surprise Disconnect of the I/O Pool Shall Be Supported	Required	Software	2.1
6.5.5	Notification of Devices Added or Removed from the I/O Pool	Required	Software	2.1
6.6	Error Handling and Telemetry			
6.6.1	Down Port Containment Support for All PNC Downstream Ports	Required	Hardware	2.1
6.6.2	Fault and Service Indicators for I/O Devices	Recommend	Hardware	2.1
6.6.3	PNC Trap of PCIe* Error Events Detected on the PCIe Link	Recommend	Software	2.1
6.6.4	Expose PNC, Device, and I/O Pooled System Telemetry	Recommend	Software	2.1
6.7	Pooled I/O System Support			
6.7.1	Device Serviceability while System Powered On	Required	Hardware	2.1
6.7.2	Pooled System Enclosure Management Support	Recommend	Software	2.1
6.7.3	AUX Power to Cable Connector	Optional	Hardware	2.1

Section	Intel® RSD Validation Criteria	Compliance	Type	Intel® RSD version
6.7.4	Exposing Cable Electrical Parameters for Cable Signal Drive Support	Optional	Software	2.1
6.8	Compute Module Requirements for I/O Pooled Systems			
6.8.1	Independent PCIe* Domain per Compute Module Connection	Required	Software	2.1
6.8.2	Down Port Containment Support for All Connected Ports	Required	Hardware	2.1
6.8.3	BIOS Shall Allocate Memory Space for All Potential I/O Devices	Required	Software	2.1
6.8.4	Compute Module Visibility of I/O Device Controlled by the PSME	Required	Software	2.1
6.8.5	Compute Module Connection Identification	Recommend	Software	2.1
6.8.6	Compute Module Managing the Assigned I/O Device	Optional	Software	2.1
6.8.7	Compute Module Managing the I/O Pool System is not Allowed	Required	Hardware	2.1
7.0	PSME Design Guidelines			
7.1	PSME Overview			
7.2	PSME Reset (Power On)	Required	Hardware	1.2
7.3	PSME Configuration Management			
7.3.1	PSME API Compliance	Required	Software	2.2
7.3.2	PSME Authentication Credential	Required	Software	2.1
7.3.3	PSME Time Sync Mechanism	Required	Software	2.1
7.3.4	PSME Telemetry Requirements	Required	Software	2.2
7.3.5	Serial-over-LAN and KVM Credential Change with User Changes	Recommended	Software	2.2
7.3.6	PSME Support for Power and Thermal Capping	Recommended	Software	2.2
7.4	PSME Software Updated			
7.4.1	PSME Remote Software Update	Required	Software	2.1
7.5	PSME Reliability, Availability and Serviceability Support			
7.5.1	Drawer Event Reporting	Required	Software	2.1
7.5.2	Drawer (PSME) Hot Add Only when RMM Is Present and Running	Recommended	Software	2.1
8	RMM Design Guidelines			
8.1	RMM Overview			
8.2	RMM Reset (Power On)	Required	Software	2.2
8.2.1	RMM Boot and PSME ID Assignment if Discrete RMM Present	Required	Software	2.1
8.2.2	RMM Assigns PSME ID if PSME Not Configured	Required	Software	2.1
8.2.3	PSME Enters "PSME ID Not Configured" State	Required	Software	2.1
8.3	RMM General Support			
8.3.1	RMM Event Handling	Required	Software	2.1
8.4	RMM Power and Cooling Support			
8.4.1	Rack Power Monitoring Support by RMM if Shared Power is Used	Required	Software	2.1
8.4.2	Rack Power Budgeting Support by RMM if Shared Power is Used	Recommended	Software	2.1
9	POD Manager (PODM) Design Guidelines			
9.2	PODM Configuration Management			
9.2.1	PODM Powered Independent of Rack Power	Required	Software	2.1
9.2.2	PODM REST API Compliance	Required	Software	2.1
9.2.3	Secure Communication Channel for Management Network	Required	Hardware	1.2
9.2.4	PODM Authentication Certificate	Required	Software	2.1
9.2.5	PODM Timestamp Support	Required	Software	2.1
9.2.6	Only One Active PODM per Pod	Required	Software	2.1
9.2.7	PODM to Allow Addition of New Drawers Only when RMM Is Alive	Required	Software	2.1
10.0	Network Switch Design Guidelines			

Section	Intel® RSD Validation Criteria	Compliance	Type	Intel® RSD version
10.1.1	Module-to-Port Mapping Configuration File Support if Dynamic Discovery Not Supported	Required	Software	2.1
10.1.2	Switch PSME Support for Base Network Services	Required	Software	2.1
10.1.3	Device Discovery and Switch Configuration Reporting	Required	Software	2.1
10.1.4	Switch Functionality Change Event Generation	Required	Software	2.1
11.0	Telemetry			
11.3.1	PSME API Support for Telemetry	Required	Software	2.2
11.3.2	PODM SB and NB API Support for Telemetry	Required	Software	2.2
11.3.3	Support for In-Band Telemetry Collection	Recommended	Software	2.2
11.3.4	Support for Correlation of IB and OOB Telemetry Data	Recommended	Software	2.2
12.0	Ethernet Pooled Storage			
12.6.3.1	Discovery Service Management	Optional	Software	2.3
12.6.3.2	Target Subsystem Management	Required	Software	2.3
12.6.3.3.1	OOB Host Configuration with Discovery Service	Optional	Software	2.3
12.6.4.1.1	Data Erase Support	Required	Software	2.3
12.6.4.2	NVMe-oF Target PSME (PSMS) Access Security	Required	Software	2.3
12.6.4.3.1	Fabric Secure Channel	Optional	Software	2.3
12.6.5.1	Volume Telemetry	Required	Software	2.3
12.6.5.2	NVMe SSD Telemetry	Required	Software	2.3
12.7.2	NVMe-oF Target Management Functional Interface	Required	Software	2.3
12.7.4	NVMe-oF Initiator Management Functional Interface	Required	Software	2.3
12.7.5	Network QOS Support	Optional	Software	2.3
12.7.6	Error Handling	Required	Software	2.3
13.0	Ethernet Pooled FPGA			2.4
13.3.1.1	Discovery Service Management	Optional	Software	2.4
13.3.1.2	Target System Management	Required	Software	2.4
13.3.2.1.1	FPGA Programming Erase Support	Required	Software	2.4
13.3.2.2	Pooled FPGA-oF Target PFMS Access Security	Required	Software	2.4
13.3.3.1	Pooled FPGA-oF target system capabilities	Required	Software	2.4
13.3.3.2	FPGA Device Telemetry	Required	Software	2.4
13.3.4	FPGA Target System Chassis Management	Required	Software	2.4
13.3.5	Hot-Plug	Optional	Software	2.4
13.4.2	Pooled FPGA-oF Target Management Interface Functional	Required	Software	2.4
13.4.3	Pooled FPGA Discovery Service Management Interface	Required	Software	2.4
13.4.4	Initiator Management Interface	Required	Software	2.4
13.4.5	Network QoS Support	Optional	Software	2.4
13.4.6	Error Handling	Required	Software	2.4

§

3.0 Intel® RSD Platform General Guidelines

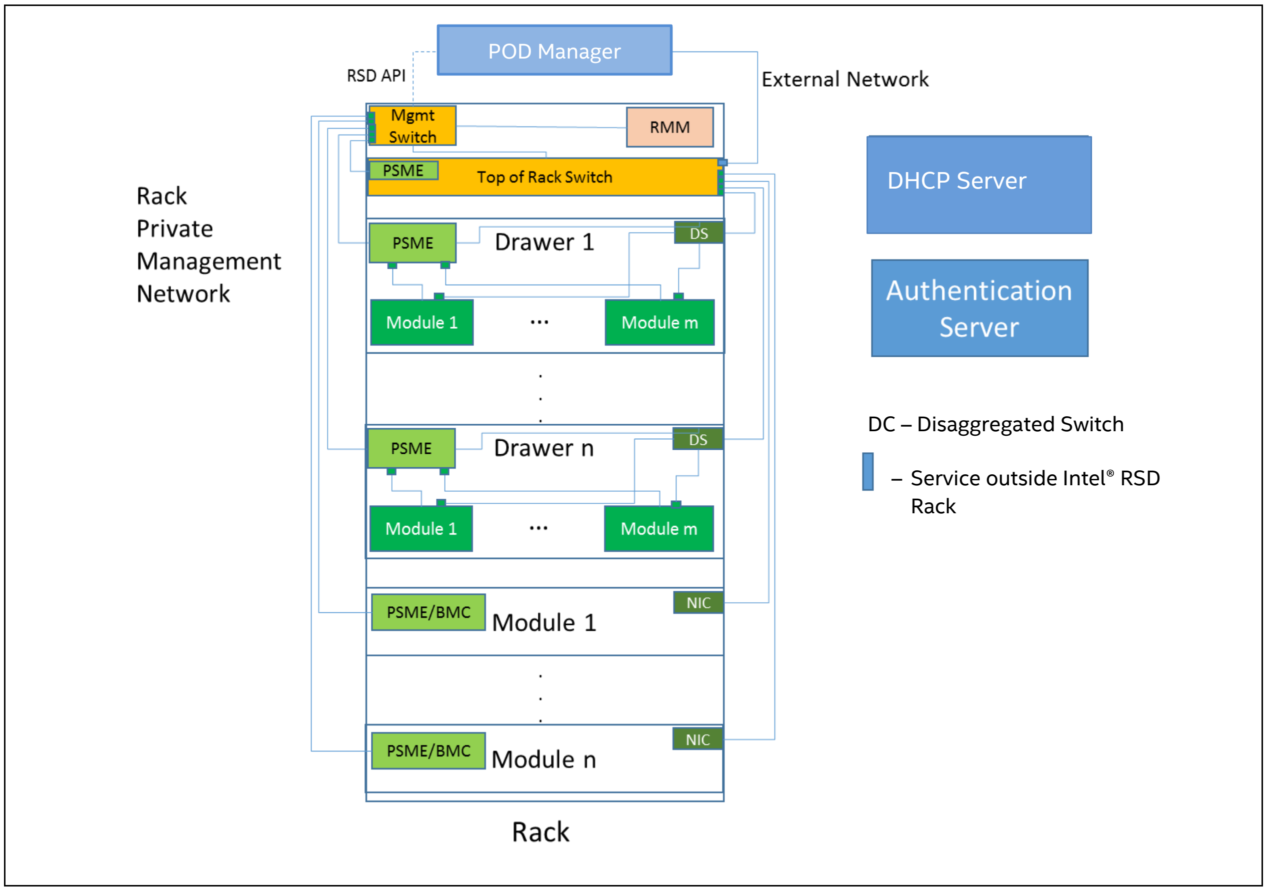
This section describes the feature design guidelines for the Intel® RSD Platform. Subsequent sections describe the design guidelines for the Intel® RSD Platform subcomponents.

3.1 Intel® RSD Platform Power-on Flow

Figure 10 shows the private Rack-wide management network for the Intel® RSD API connections in a POD.

Note: In Figure 10 the term "network" refers to the Ethernet fabric connection and the term `"management plane"` refers to the management control or function between management agents.

Figure 10. Management Plane in the Intel® RSD Rack



All Racks in the POD are powered on when power is applied. Applying power to each Rack starts the Rack Management Module (RMM), PSME, and TORS resources in each Rack. Once power is applied, the PODM can communicate with the RMM and PSMEs in the Racks to collect the hardware configuration information from the managed resources.

The POD must step through a sequence of events to create one (or more) Composed Nodes. The event sequence is as follows:

1. Assumptions:
   - POD has access to the data center Network Time Protocol (NTP) server.

- If host names are used for the PODM, RMM, and PSME, then the data center provides access to the Distributed Host Configuration Protocol (DHCP) and Domain Name System (DNS) server.
- RMM and TORS need manual configuration.

2. Prerequisites:
   - If the DHCP and DNS are not available, provision the PODM, RMM, and PSME with static IP addresses.
   - Provision the RMM with PODM authentication credentials and PODM host name or PODM IP address.
   - Configure switches (such as TORS) to allow the RMM and PSME to connect to the data center Out-of-Band (OOB) management network.
   - The SSH may be used to provision the above steps.
3. Apply power to the POD:
   - Power is applied to all the Racks in the POD:
     – All the TORSs are powered.
     – All Rack Management Controller (RMC) and Drawers are powered, and RMM firmware starts execution.
     – All the Drawers are powered, and PSME firmware starts execution.

Note: When power is applied to Drawers (in Step 3), the sequence of power applied to the Modules, Blades, and PNC is implementation-specific, as controlled by the PSME.

4. The RMM and PSME are assigned IP address:
   - If the host name is used, then the IP address is obtained from the DHCP server. The DNS maintains the latest host-name-to-IP address mapping.
   - If a static IP address is used, the DHCP server is not required.
5. The RMM and PODM create a Transport Layer Security (TLS) session with two-way authentication[1]. Although the hard requirement is the RMM authenticates the PODM, the RMM is acting as the TLS server.
6. PODM registers the RMM into its configuration management database (CMDB), with the status "RMM initialization started."
7. PODM waits for the RMM to be ready to service requests
   - The RMM is ready to receive the queries from the PSME through the Rack-wide private network.
   - The RMM discovers the power supply and the fans, and starts managing them.
   - The RMM finishes its initialization and is ready to service requests from the PODM.
8. PSME gets the IP address from DHCP, Rack ID, and PODM credentials from the RMM using the Rack-wide private network.
9. PSME and PODM create a TLS session with two-way authentication, and establish a secure communication channel with PODM.
10. PODM registers the PSME into its CMDB, with status "PSME initialization started."
11. PSME discovers the Drawer inventory (such as Modules and Blades) by communicating with the Module Management Controller (MMC) and BMC.

Note: This step can happen in parallel while the PSME is establishing communication with RMM and PODM.

12. PSME presenting presence through the Simple Service Discovery Protocol (SSDP) denotes PSME initialization stage complete.
13. PODM requests a "Drawer inventory" from each PSME.
14. Repeat Steps 11 through 16 for each PSME in the Rack.
15. At this point, the PODM is ready to create and assign Composed Nodes.

[1] *Although the hard security requirement is for the RMM/PSME to authenticate the PODM, not the other way around, because RMM/PSME is the server for the TLS connection, it's necessary for two-way authentication. For cases where RMM/PSME authentication is not required i.e., a data center doesn't want to incur the cost of provisioning certificates for all RMMs/PSMEs, then the PODM could accept a self-signed certificate from RMM/PSME.*

Note: The rack private management network could be physically or logically separated. This document uses physical separation as a base design to describe the requirements.

3.2 Generic Intel® RSD Platform Requirements

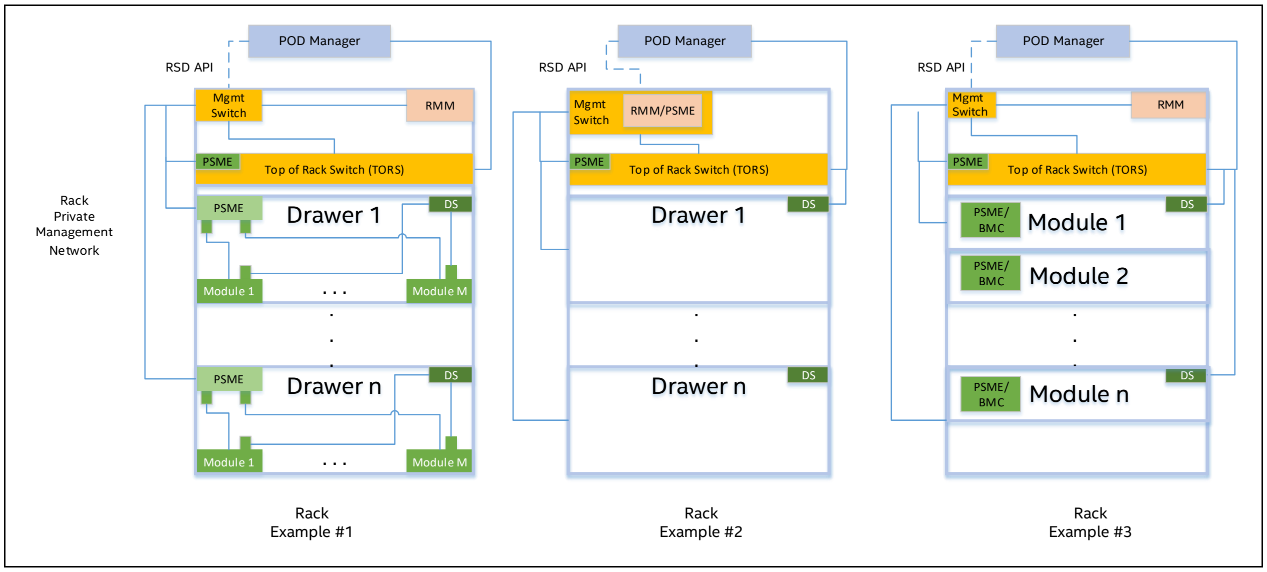
The Intel® RSD Platform meets the following generic requirements as stipulated (optional, recommended, or required) in each section as shown in Table 4.

3.2.1 Rack Must Have One or More Logical Pooled System Management Engine (PSME) Software

Required:

Figure 11 shows three examples of Intel® RSD architecture implementations. In example 1, the rack consists of drawers within the PSME; the PSME interfaces with the modules and it has a separate RMM module. Example 2 implementation, the RMM and PSME reside on the same hardware, and example 3 implementation consists of modules with a BMC that exposes the PSME API. Figure 11 shows all possible implementations, but the key point is that a logical RSD rack must have one or more PSMEs.

Figure 11. PSME in a Rack



3.2.2 Shared or Highly Efficient Power Supply

Required:

For optimal TCO, Compute/Storage Modules must support cost-effective, efficient, and manageable shared power. The solution is achieved by either:

- Sharing power across two or more Modules, or
- Having a > 90 % efficient (delivered power to Module/input AC power) power configuration.

3.2.3 Shared or Highly Efficient Cooling

Recommended:

For optimal TCO, Compute/Storage Modules are recommended to support cost-effective and manageable shared cooling. The Intel® RSD Platform is recommended to support shared cooling across two or more Modules. One option for shared cooling is to use a fan larger than 2u (3.5 inches) in diameter. If the system does not use fans, another (more efficient) cooling option is to implement shared liquid cooling.

Note: If shared power is used, then a shared cooling solution is recommended.

3.2.4 Just a Bunch of Disks (JBOD) Support

Optional:

Just a Bunch of Disks (JBOD) is a collection of storage devices in a chassis for easy serviceability. The JBOD is generally connected through a cable (such as Serial ATA* (SATA*), PCIe*, and so on) between a Compute Module and a Storage Module.

3.2.5 Compute Module with Local Boot Drive

Optional:

The Intel® RSD Compute Module has an Internet Small Computer Systems Interface (iSCSI) or Preboot eXecution Environment (iPXE) support which makes the boot or storage services available through the network. Compute Modules could use local storage such as an M.2 drive, Solid State Drive (SSD), or HDD for boot or for delta-file storage to achieve better performance.

3.2.6 At Least One Intel® RSD Compute Module in a POD

Required:

To compose systems, compute, and workloads, the RSD POD must include at least one Compute Drawer with at least one Compute/Storage Module. It is possible that some racks support only storage and some support only compute modules, but a POD must contain at least one compute module.

3.2.7 Compute Module Serviceability Independence

Required:

To keep data centers always up and running and achieve hyper scale-like agility, Compute/Storage Modules must support modular CPU and memory resources that can be serviced or upgraded independent of other modules. The RSD Modules must be hot-pluggable into the RSD Drawer without requiring the Drawer to be powered down to provide high- RAS for Intel® RSD platforms.

3.2.8 Ethernet-Based Fabric for Management and External Network Connectivity

Required:

The Intel® RSD Platform must support Ethernet-based fabric for management and external network connectivity.

Note: An Intel® RSD platform will support access to (at least) two isolated networks, one for normal in-band traffic, and one for OOB management.

3.2.9 At Least One Ethernet Switch in the POD

Required:

The Intel® RSD POD must have at least one Ethernet switch component that connects the POD to the external network.

3.2.10 Network Switch Support for Network Software Agent

Required:

Network switch components (such as TORS) must support running a network software agent to configure network switches. The required switch management capabilities are described in the *Intel® RSD PODM* and *Intel® PSME API specifications*, Table 3, must be implemented.

3.2.11 PODM Support for PNC Capabilities

Recommended:

The Pooled Node Controller (PNC) provides connectivity of pooled I/O devices through a PCIe fabric to compute modules. The PNC is managed by a PSME which allows the PODM to assign I/O devices in the pool to a Compute Module. Refer to Section **Error! Reference source not found.** for more details. If the PNC is supported in the rack, hen the PODM must support PNC PSME API compatibility.

3.2.12 Platform Support for FPGAs

Recommended:

It is strongly recommended the platform is capable of supporting the FPGA in the platform. The use of an FPGA has been shown to be a key performance differentiator in both machine learning and big data processing workloads. If an FPGA is present, the PODM, and PSME must discover these capabilities as described in the Intel® RSD API specifications. Refer to Table 3 for a list of specifications.

3.2.13 Hot-Pluggable Modules in Intel® RSD Drawers

Required:

In a Hyper Scale datacenter where uptime is required at all times, services cannot be disrupted by adding or removing modules. The Intel® RSD Modules must be hot-pluggable into the RSD Drawer without requiring the Drawer to be powered down to provide high-RAS for Intel® RSD platforms.

3.2.14 PDOM Backward-Compatibility for Intel® RSD v2.3 and Above

Required:

To achieve optimal TCO and improve overall data center operation, as well as ease the adoption of newer Intel® RSD versions, Intel® RSD v2.3 must support backward compatibility with previous generations of Intel® RSD v2.2 and Intel® RSD v2.1 racks. In Intel® RSD v2.3, the storage services APIs moved from proprietary APIs to standard based Swordfish* APIs. To deal with storage service backward compatibility, either PODM v2.3 needs to provide a storage proprietary API to the Swordfish API or the existing storage PSME needs to be updated to the Swordfish API. Intel® RSD v2.3 PODM implemented the latter and hence the Intel® RSD v2.3 PODM does not provide backward compatibility for Intel® RSD v2.1 and v2.2 storage services.

3.2.15 Drawer Backward-Compatibility for Intel® RSD v2.3 and above

Required:

To achieve optimal TCO and improve overall data center operation, as well as ease the adoption of newer Intel® RSD systems into existing data centers and maintain compatibility while updates are happening, a PODM needs to be able to manage both a PSME v2.2 and v2.3 existing in the same rack.

If Intel® RSD v2.2 and v2.3 Drawers are mechanically compatible the user should be allowed to interchange Drawers.

3.2.16 Intel® RSD Versions Coexistence Support within a Rack

Required:

Every generation of Intel® RSD introduces a new set of features. These new features are introduced with standard body support, such as DMTF Redfish*, Swordfish*, or with Intel® RSD proprietary APIs. The Intel® RSD proprietary APIs are generally replaced by standard based APIs in future Intel® RSD versions. Table 5 lists general expectations for various Intel® RSD versions backward compatibilities.

Table 5. Intel® RSD Component Versions and Platform Support Matrix

PODM	RMM	PSME
1.2	1.2	1.2
2.1	1.2 and 2.1	2.1
2.2	1.2, 2.1, and 2.2	1.2 and 2.1
2.3	2.1, 2.2, and 2.3	2.3 2.2 and 2.1 (Compute PSME only)
2.4	2.4	2.4

3.2.17 PODM-to-PSME Communication Channel Protection

Required:

To withstand network attacks, secure the PODM, and PSME communications. To provide this security, protect the communication channel between the PODM and PSME by the use of TLS.

3.2.18 PODM-to-RMM Communication Channel Protection

Required:

To withstand network attacks, secure the PODM, and RMM communication. To provide this security, communication between PODM and RMM must be protected by using a private network or an OOB management network.

3.2.19 PSME-to-RMM Communication Channel Protection

Required:

To withstand network attacks, PSME and RMM communication must be secured. To provide this security, PSME and RMM communication is protected by using a rack private network that is not reachable from outside the rack.

3.2.20 In-Band Reconfiguration of the Private Network

Required:

To withstand network attacks and have a more secure design, the in-band software (application or workloads) on the Compute Module must not have access to the OOB management interconnect (i.e., rack private network or OOB management network).

3.2.21 User-Maintained Backup Copy of Data

Recommended:

Before a user returns an unneeded Compute/Storage Module resource to the PODM for re-composition, the user is responsible for backing up any data stored in the local memory on the Module. Once the PODM re-composes the resources, data stored on a Compute/Storage Module by the previous user will be cleared.

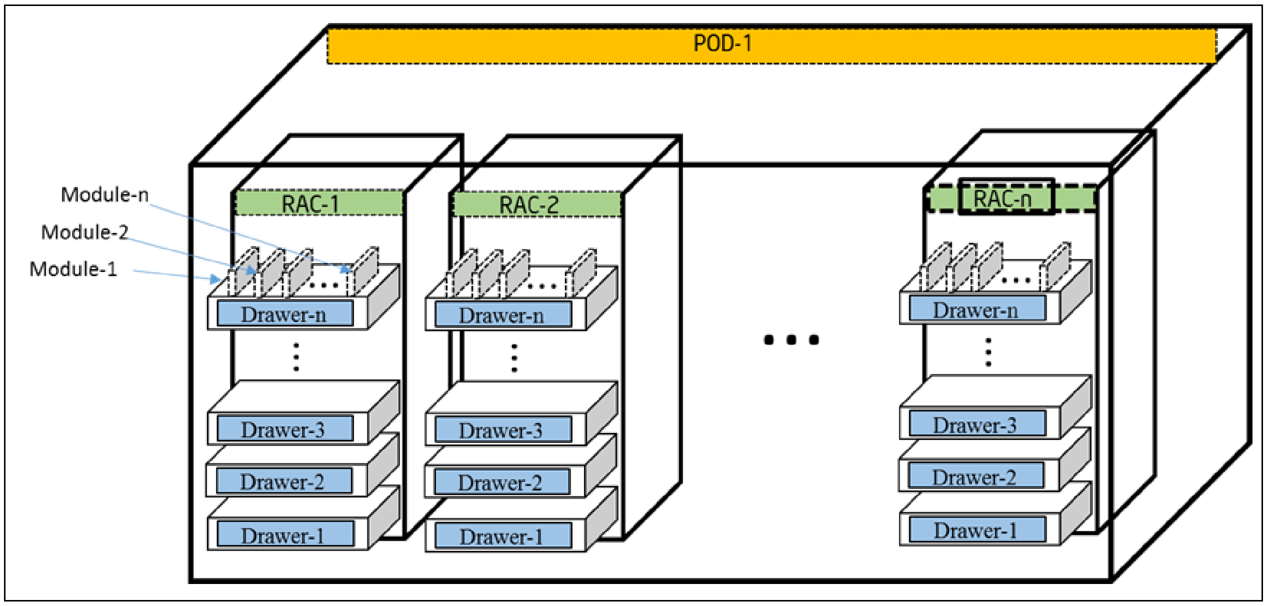
Note: By default, the data must be secure erased. It is strongly recommended that NVMe drives with secure erase capability and used for storage.

3.3 Intel® RSD Components Location Identification Support

To improve overall datacenter operations, a key attribute of Intel® RSD management is location-aware discovery. A data center manager or maintenance technician should be able to identify the physical location of RSD hardware components so they can be serviced. Figure 12 illustrates an example POD and RSD hardware components across multiple Racks.

Refer to Table 3, Intel® RSD API specifications for data format details.

Figure 12. Intel® RSD v2.1 Component Location Identification



3.3.1 Field Replaceable Units Identification and Location Information

Required:

To help a service representative locate and identify the Field Replaceable Unit (FRU), all Intel® RSD components that are reported as FRUs must provide a unique identification number and location of the FRU. Refer to the Intel® RSD API specifications (Table 3) for the detailed format of component IDs and location IDs for supported components. The System Management BIOS (SMBIOS) records generally provide the location of the component within the module. The RSD API must provide the location of the module within the chassis and the location of the chassis within the rack. In some cases, system admin could assign the location ID information chassis, such as Rack Chassis, if supported it must follow the Intel® RSD API specifications listed in Table 3.

3.3.2 Connectivity Identification

Required:

The PODM must be able to understand the hardware connectivity topology of its components to provide better manageability and serviceability features. For example, compute node to storage node port/drive connectivity information helps to highlight the error path in case of a compute node storage access failure. Refer to Table 3, Intel® RSD API specifications for APIs to represent connectivity.

3.4 Intel® RSD Fabric and Network Configuration

3.4.1 OOB Management Network and In-Band Data Network Separation

Required:

For a more secure design, the Intel® RSD Platform must support access to two separate networks: one for OOB management access and one for in-band data access for the host.

3.4.2 Secure NTP Access Availability

Required:

To improve overall data center operations, RSD event logs must be time stamped to help determine the event generation sequence. Intel® RSD system components (such as the PODM, RMM, and PSME) must log events and errors. Intel® RSD components must synchronize time periodically from the secure Network Time Protocol (NTP). This helps put together an overall POD-wide event list and helps reduce the amount of time required to identify root cause issues. If only the PODM has direct access, then the RMM and PSME must be able to get the time from PODM.

3.4.3 Secure DHCP Server Availability if DHCP Discovery Is Used

Recommended:

If dynamic addressing is used with the PODM, RMM, and PSME using DHCP, then it is recommended that the data center implement a secure DHCP server to facilitate IP address assignment for the PODM, RMM, and PSME. If static IP addresses are assigned to the PODM, RMM, and PSME, then a DHCP server is not needed for POD management. Another option is to use Simple Service Discovery Protocol (SSDP) to discover and assign IP address.

3.4.4 Secure DNS Support

Recommended:

If host names are assigned to the PODM, RMM, and PSME, then it is recommended the data center implement a secure DNS service to facilitate collecting the IP addresses for the PODM, RMM, and PSME.

3.5 Intel® RSD Platform Configuration and Provisioning

This section describes the admin control, configuration and provisioning required to troubleshoot and debug issues remotely, prevent tampering of the firmware and unauthorized access to the datacenter.

3.5.1 Serial over LAN (SOL) or KVM Support for Compute Modules

Required:

To troubleshoot and debug issues remotely for improved overall datacenter operations, Compute Modules must provide serial console redirection support over LAN or keyboard, video, and mouse (KVM). Generally, this service is supported through a dedicated BMC or through a shared BMC from compute node.

3.5.2 Firmware and Software Updates Signed and Checked

Required:

For a more secure design, and to prevent tampering of the firmware, all Firmware (FW)/Software (SW) updates must be signed, and the integrity of the updates must be checked before they are used by individual components.

3.5.3 PODM or Admin Control of Updates

Recommended:

For a more secure design and to prevent unauthorized access and tampering, it is recommended that a request for the FW/SW update should only come from the PODM or an admin logged on to the hosting Platform who has the appropriate access rights.

3.6 Intel® RSD Platform Security

This section contains the Intel® RSD platform security architecture. Not all components specified in this section are available in the current Intel® RSD reference. Refer to Table 3, Intel® RSD API specs for security features implemented in each Intel® RSD revision reference implementation.

3.6.1 Intel® RSD Platform Security Architecture Overview

Security work for the Intel® RSD Platform is driven by the following guiding principles:

- An RSD Composed Node provides at least the same level of security as an equivalent Intel® Architecture (IA) standalone server.
- An RSD cluster (made up of a set of Composed Nodes) is at least as secure as an equivalent cluster built with standalone IA servers.

Composed Nodes in the Intel® RSD system may be a standalone IA server or "assembled" (composed) from disaggregated resources, and in some cases from shared components. Specifically, Intel® RSD systems use management software at the POD, Rack, and Drawer levels to manage the inventory of components used to compose server Modules, to enable server Modules once they are composed, and to provide telemetry at different levels.

To follow the guiding principles above, Intel® RSD Platforms must support the following security objectives:

- Maintain integrity and availability of the Intel® RSD Platform
- Maintain isolation between (workloads running on) Composed Nodes in the presence of shared components

3.6.1.1 Maintain Integrity and Availability of the Intel® RSD Platform

To achieve this objective, RSD must provide the means for ensuring the following:

- Installation and update time integrity protection of all Intel® RSD management software (SW) and firmware (FW)
- Runtime protection of the Intel® RSD management infrastructure
- Support for datacenter administrator separation of duties and time of update
- Protection against permanent denial of service (PDoS) as a result of a cyberattack

3.6.1.1.1 Installation and Update Integrity Protection

This requirement requires every FW and SW element inside the Intel® RSD trust boundary must be authenticated at installation and update time. Authentication in this context means there must be a way to verify that software/firmware being installed has not been changed from what was originally delivered by the author.

To allow for authentication of pre-built PSME/PODM packages, they shall be signed with an Intel certificate.

> For users who compile the code and build packages in their own environment, it is recommended that they sign certificates with GPG keys for safe distribution.

Refer to the *Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) User Guide,* Appendix E.12 and *Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) Representational State Transfer (REST) User Guide* Section 2.9, Table 3, for a description of the process of creating keys and signing packages.

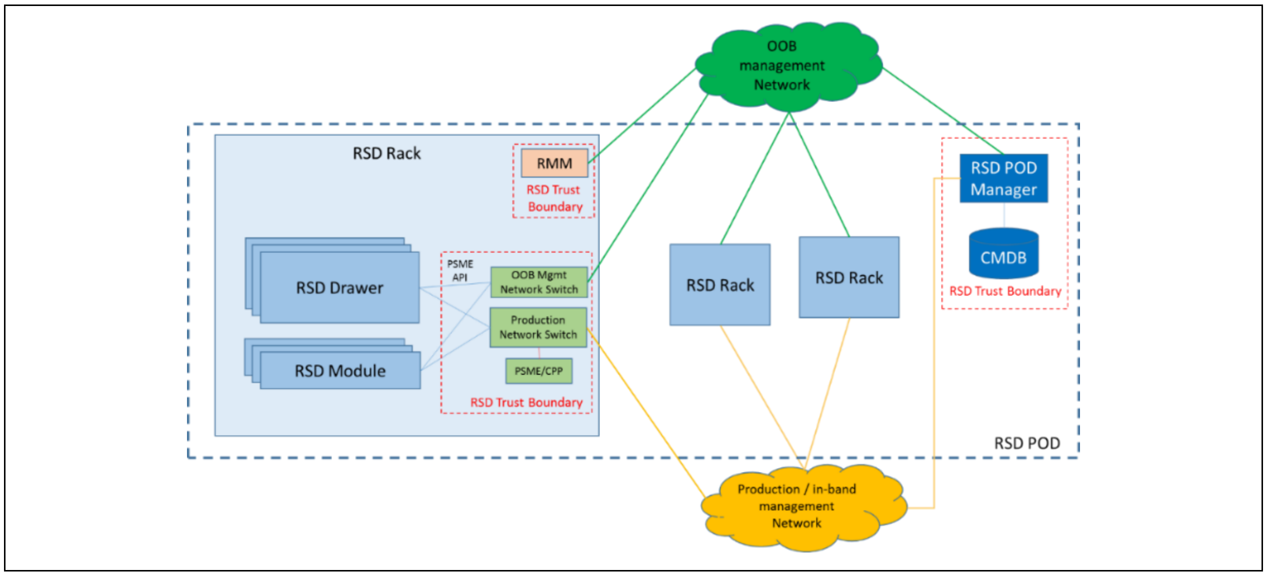
For additional protection, all PSME/PODM packages must generate SHA512 checksums of the executables upon installation.

If an executable was tampered with, a service shall detect that the checksum doesn't match and refuse to start.

3.6.1.1.2 Runtime Protection of Intel® RSD Management Infrastructure

This requirement refers to ensuring that operation of the Intel® RSD management infrastructure is protected from entities that are outside the RSD trust boundary as outlined in Figure 13.

Figure 13. Intel® RSD Trust Boundary



The Intel® RSD management infrastructure must be protected from entities outside the RSD trust boundary, including any management entities connected to the OOB management network, including other RSD PODs, other data center management servers, networking gear, and so on. Any software running on composed nodes or storage bricks that are made available through the Intel® RSD management infrastructure, and any other external entities that may be connected to other datacenter networks.

The main tools for achieving runtime protection of the Intel® RSD management infrastructure are isolation and access control.

Regarding isolation, users must ensure that logical management functions either run on dedicated hardware or that the hardware and system software hosting the Intel® RSD management logical functions provide this isolation from whatever other services they may be hosting. For example, if a single server is hosting both the POD management function and an independent datacenter switch management function, the system software in that server must ensure the switch management function cannot be used to compromise the POD management function. In this particular case, the POD management function and the system software on that host would be inside the RSD trust boundary, but the switch management function would be outside. The system software must enforce this on that host.

Intel® RSD shared resources (i.e., a network switch) must also enforce isolation between their different users. In the case of the switch, it must implement isolation between the OOB management network and the other networks in the data center. Enforcing this isolation means, that software running on composed nodes will have access to the orange networks, but not to the green network. (Refer to Figure 13.) This isolation is critical for providing an environment in which OOB management functions can be isolated from the rest of the datacenter and addresses the requirement for protecting the Intel® RSD management infrastructure from elements outside the Intel® RSD trust boundary that have access to production in-band data networks, including software running on composed nodes.

To support early initialization, security credential provisioning, and protect communications between elements inside the rack that do not support adequate access control and should not be visible outside the rack, RSD racks must support a private and self-contained network (i.e., not reliant on external services like DHCP, DNS, and so on). This private network is not accessible outside the rack.

Note: Because the different elements inside the RSD trust boundary do not reside on the same rack but must communicate with each other using an OOB management network that is outside the RSD trust boundary, isolation is not enough to guarantee runtime protection. To allow protected communication between the different elements inside the RSD trust boundary, it is necessary to use access control.

For Intel® RSD, access control means protecting all APIs provided by Intel® RSD logical management functions, as well as protecting logon access to all hardware hosting those functions.

Access Control for Intel® RSD APIs:

PODM:

PODM APIs are REST-based and must be protected using Hyper Text Transfer Protocol with Secure Sockets Layer (HTTPS). Authorization and authentication of users for these APIs is datacenter-specific.

RMM:

The RMM handles rack level infrastructure functions like power and cooling, as well as provisioning of information (including security credentials) required for early rack initialization.

RMM provides a set of REST-based APIs that allow a PODM to interact with it; these APIs are protected by HTTPS. Access control is such that only the controlling PODM can access these APIs. In other words, the PODM must authenticate itself to the RMM. If this authentication is successful, a secure communication channel (for example, TLS based) is established between the PODM and RMM.

PODM authentication relies on a credential that is provisioned to the RMM at the time of rack deployment. This credential will be distributed by the RMM to the PSMEs in the rack during rack early initialization, or when a new PSME is powered on in an existing rack, using the rack private network.

PSME:

The RMM and PSME provide a set of REST-based APIs that allow the PODM to interact with them; these APIs are protected by HTTPS. Only the controlling PODM can access these APIs for Access Control. In other words, the PODM must authenticate itself to the PSME. If authentication is successful, a secure communication channel (for example, TLS based) is established between PODM and PSME.

The credential used for PODM authentication is provisioned by the RMM during early rack initialization, or when a new Drawer is first powered on in a rack, using the rack private network.

Role-Based Authorization

Optional for Intel® RSD v2.3:

Intel® RSD Security architecture allows for role-based authorization. The supported roles are RSD Viewer, RSD Admin, RSD Network Admin, and RSD Global Admin.

PODM will enforce access control to Intel® RSD functionality by external (to RSD) datacenter management infrastructure based on the roles described in Table 6.

Table 6. Intel® RSD Admin Roles

Role	Access mechanism	Authorization mechanism	Restrictions
RSD Viewer	HTTPS	DC specific	HTTP GET only (read-only access), no ability to change state of any RSD element
RSD Admin	HTTPS	DC Specific (for example, OAuth)	Can access most PODM APIs and effect changes. Cannot access switch configuration and management APIs?
RSD Network Admin	HTTPS	DC specific (for example, OAuth)	Can only access switch configuration and management APIs.
RSD Global Admin	HTTPS	DC Specific (for example, OAuth)	Can access all PODM APIs.

3.6.1.1.3 Access Control for Hosts Inside the Intel® RSD Trust Boundary

Intel® RSD does not dictate a one-to-one relationship between an Intel® RSD management logical function and its physical host. The same physical host could host multiple functions.

It may be possible to interact with the RSD by logging into the physical hosts (through a local console or SSH) of the RSD logical management functions. Login access to these hosts should be strictly controlled as much as possible; access should be restricted to a system admin of the physical host. Although not recommended from a security perspective, a physical host may host more than just the RSD logical management functionality[2]. In this case, responsibility for ensuring that non-RSD functionality cannot interfere or compromise the operation of Intel® RSD functionality, falls entirely on system software and configuration of that host. Intel® RSD logical management functions have no way of protecting themselves from privileged level software or privileged users (for example, root) in the physical host.

For the purpose of this discussion, we will assume that their own physical hosts host the PODM, RMM, MMC/BMC, and its own physical host hosts the PSME (in a given Drawer).

Table 7. Login Access to Intel® RSD Management Hosts

Logical Management Function	Physical Host	Users (logon)	Default Credentials
PODM	DC admin server (local console and `.ssh`)	Host admin(root) Other non-privileged non RSD users (optional)	User name and password
RMM	Rack controller (local console and `.ssh`)	RMM Admin (root)	User name and password
PSME	Drawer controller (local console and `.ssh`)	PSME Admin (root) Network admin (no access to PSME functionality)	User name and password
MMC/BMC	Baseboard management controller (local console. local console and `.ssh`)	BMC admin (root)	User name and password

Note: Table 7 summarizes the different logon accesses for the Intel® RSD logical management function hosts. Each of these hosts supports an admin logon, which is used for managing the host and for installing software on it. This user is entirely in the RSD trust boundary as it has full control of the host and its software.

3.6.1.2 Support for Administrator Separation of Duty

Intel® RSD supports admin separation of duty by restricting access to RSD management functionality only to admins with the appropriate RSD admin roles.

Intel® RSD logical management functions must provide protected logging functionality that will help in forensics work.

3.6.1.3 Permanent Denial of Service

Optional for Intel® RSD v2.2 or later versions:

Intel® RSD platforms must protect themselves from PDoS that could result from cyberattacks. Specifically, this means that Intel® RSD platforms must have the ability to recover without requiring administrator physical presence or factory involvement.

Security requirements for supporting automated recovery from cyberattack include:

[2] This may be the case for the PODM physical host, but don't expect it to be the case for other RSD logical management function hosts.

- The platform must define a minimum set of FW and/or SW that is critical to enabling platform-remote recovery. The platform must ensure this firmware is always available[3].
- The platform must be able to recover critical FW or SW compromised during an attack without requiring special equipment or physical presence by an administrator.

3.6.1.4 Maintain Isolation between Composed Nodes

Intel® RSD platforms are primarily targeted at cloud data centers and multi-tenant environments. As such, they must provide strong isolation between workloads running on different composed nodes.

Security requirements for supporting strong composed node isolation include:

- Intel® RSD platforms must ensure node isolation in the presence of shared physical resources (for example, storage, memory, network, and so on).
  - Software (even at the highest privilege) running on a composed node must not be able to access shared resources allocated to another node.
  - Intel® RSD management SW must be protected from the composed nodes only performs configuration of shared resources.
  - Shared resources must support access control to ensure that authorized composed nodes can only access them.
- The Intel® RSD platform must prevent nodes from interfering with other nodes sharing a given physical resource (i.e., support basic Quality of Service (QoS) for shared physical resources).

3.6.1.5 Support Composed Node Attestation

Optionally introduced in Intel® RSD v2.2:

The Intel® RSD platform must support attestation of composed nodes, which is a way for a data center to prove to a workload owner (remote verifier) that the platform in which that workload is running supports the security needs of that workload.

Specific security requirements for supporting attestation includes:

- The Intel® RSD platform must provide a hardware root of trust for measurement (RTM) for Composed Nodes and for RSD logical management function hosts.
- Each composed node in the platform must support providing attestation evidence about the environment in which that node was started.

3.6.1.6 Intel® RSD Private Rack Management Network

The Intel® RSD private rack management network is a key element of Intel® RSD security. This scheme requires RSD management elements such as PSME, RMM, and Storage Node BMCs to contain at least two logical networks, where one logical network is attached to the rack-wide private management network and the other is attached to the datacenter's out-of-band management network.

3.6.1.6.1 DHCP Server

The DHCP server assigns external IP address to the RSD management elements (PSME, RMM, Storage Node BMC...). DHCP server also provides information to PODM about the identity of the RSD management elements (PSME, RMM, and so on). Further, the DHCP server response also contains the IP address of the RMM if the rack contains an RMM.

[3] *This does not necessarily mean that this firmware cannot be compromised (although this is a way to make it always available!), but if compromised, the platform has the ability to replace it with a good copy without having to rely on admin intervention.*

As such, the DHCP server is in the trust domain and serves as one of the root of trusts on which Intel® RSD Security is built. Since DHCP is broadcast based, it is the responsibility of the environment where RSD is deployed to protect against a rogue (either malicious or accidental) DHCP server(s).

3.6.1.6.2 Intel® RSD Drawer Manageability Security

Intel® RSD Drawers may have other manageability engines, such as Compute Node BMC. RSD Drawer implementation is required to ensure that PSME is able to:

- Securely communicate with these manageability engines (no threat of spoofing or replay attacks)

  Ensure that PSME is in a master role over these manageability engines (for example, if the BMC implemented a LAN-based access mechanism, then the access security is configured so that PSME is the only master allowed to effect changes).

3.6.2 Composed Node Volatile Memory Clearing

Required:

The composed node reallocation needs to prevent a new user from accessing previous user contents. If the Composed Node resources are de-composed and reallocated, then the contents of volatile memory must be cleared before the resources are reallocated (one-way to do this is to perform a cold reset on the Compute/Storage Module).

3.6.3 User to Archive Data before Decomposing a Node

Recommended:

If a user wants to preserve the persistent data in memory, before decomposing a Composed Node, it is recommended the user store their data in a remote storage area before relinquishing control of the resources in the Composed Node. After decomposition the memory will be returned to the pool of resources for reuse.

3.7 Intel® RSD Power and Cooling

3.7.1 Power Monitoring Support

Required:

Intel® RSD must support power monitoring at the rack, Drawer, and module level. This feature helps the user to determine if a Drawer or module exceeds a certain power budget level, and to take action to stay within the rack power limit. If the BMC exposes/supports Intelligent Platform Management Interface (IPMI), it is strongly recommended to follow specification *Intel® Intelligent Power Node Manager 4.0 External Interface Specification Using IPMI,* Doc #332200, refer to Table 3 or later version, or functional equivalent.

3.7.2 Power Budgeting Support

Recommended:

Intel® RSD recommends using a power control logic for the BMC/PSME to limit the power to the nodes. If the BMC exposes/supports Intelligent Platform Management Interface (IPMI), it is strongly recommended to follow the *Intel® Intelligent Power Node Manager 4.0 External Interface Specification Using IPMI,* Doc #332200, refer to Table 3 or later version, or functional equivalent.

3.7.3 Cooling Failure Reporting

Required:

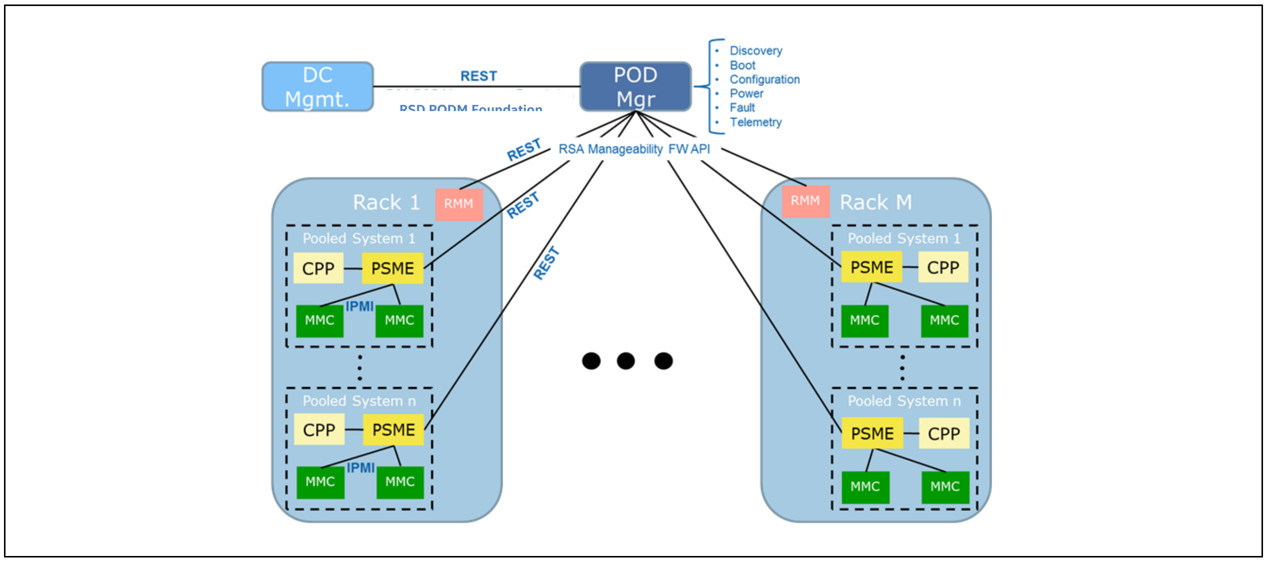
For improved data center operations, the location and status (on/off/failures) of a fan must be reported so that maintenance actions can be performed. An RSD system allows fans to be placed at Rack, Drawer, or the Module levels, or in combinations, or have liquid cooling. This must be implemented as described in Intel® RSD API specifications. Refer to Table 3.

§

4.0 Intel® RSD API

Intel® RSD management software interfaces with PODM, RMM, and PSME using the Intel® RSD API (PODM REST API RMM REST API, and PSME REST API specifications), as shown in the block diagram in Figure 14. This section outlines the top-level API requirements. Refer to the Intel® RSD API Specifications, Table 3, for API details and individual parameter requirements for each API.

Figure 14. Intel® RSD v2.3 API Block Diagram



4.1 Intel® RSD API Interface

Intel® RSD APIs are supported by the PODM, RMM, and PSME. The Intel® RSD API uses the REST protocol. The REST protocol is built using HTTP and HTTPS.

4.1.1 Intel® RSD API Compliance

Required:

The PSME and RMM NB API must support the *Intel® RSD API Specification*. Refer Table 3, Intel® RSD API specifications for the Intel® RSD Schema for required, recommended, and optional parameters for various Intel® RSD APIs. All Intel® RSD components such as PODM, RMM, and PSME are required to comply with Intel® RSD API definitions.

4.1.2 Intel® RSD API Support for Access Control and Secure Communication Channel

Required:

All Intel® RSD APIs must support access control and secure communication channels except the root entry point.

§

5.0 Module Design Guidelines

This section describes the Intel® RSD Platform Compute/Storage Module design guidelines. A Compute/Storage Module is generally a combination of compute and storage resources with network and/or storage connectivity. In some cases, the terms Module and Blade may be a used interchangeably for a single hardware element.

5.1 Module Reset, Power, and Performance

5.1.1 Module Power On/Off Support

Required:

For the ability to conserve power when a module is not in use, the PSME must provide power on/off support for each Compute/Storage Module.

5.1.2 Module Reset Support

Required:

For dealing with configuration and software changes, the PSME must provide reset support for each Compute/Storage Module.

5.1.3 Power Monitoring Support

Required:

For improved TCO and high availability of services, it is required that Compute/Storage Modules implement power monitoring support and support rack level power monitoring. Refer to Section 3.7.1 for related details.

5.1.4 Power Budgeting Support

Recommended:

For improved TCO, datacenter operations and high availability of services, it is recommended that Compute Modules support power budgeting. Refer to Section 3.7.2 for related details.

5.2 Module Features

5.2.1 Expose TPM Capabilities if TPM Present

Required:

If TPM is present as a physical component on the motherboard, or embedded firmware in ME, the PSME must expose the TPM version number and capabilities. Refer to Table 3, Intel® RSD API specifications for interface API information.

5.2.2 Expose SMBIOS Information to PSME

Required:

The Intel® RSD Compute Module must expose SMBIOS information to the PSME for it to expose the module capabilities to the PODM as defined in the *Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) Representational State Transfer (REST) API Specification*, Table 3.

5.2.3 Expose FPGA Capabilities if FPGA Is Present

Required:

If the FPGA is present in the compute module (integrated into either the processor socket or discrete module in the board), its Accelerator Function Unit (AFU) capabilities must be exposed through the OOB for the PSME. Refer to Table 3, Intel® RSD API Specifications for details.

5.2.4 BIOS/Firmware Support for PNC if PNC Supported

Required:

The rack that supports the PNC compute and PNC modules must provide a mechanism to verify cable connectivity. If the cable does not have a clock, then Separate Refclk Independent SSC (SRIS) should be enabled to allow the compute module to handle link failures.

5.2.5 Minimum 10 GbE Network Interface Card (NIC) per Module for Data Plane

Recommended:

To keep up with workload networking demands in a hyper scale-like data center, it is recommended that a Compute/Storage Module support a logical connection to one port with a minimum 10 GB Ethernet Network Interface Card (NIC) speed.

5.2.6 Expose the Presence and capability of Intel® Optane® DC DIMMs

Required:

If Intel® Optane® DC DIMMs are present in the module system memory array, the capabilities, mode of operation, and capacity must be exposed through the OOB for the PSME. Refer to Table 3, Intel® RSD API Specifications for details regarding these devices.

5.3 Module Firmware Update

Compute/Storage Module firmware updates are generally performed as part of a normal maintenance cycle for bug fixes or upgrading to a new feature set. Critical updates are typically performed for fixing security-related issues, important bugs, and even new firmware versions containing new capabilities. An in-band firmware update operation is time-consuming and less flexible for management. An RSD aligned Compute/Storage Module must provide the capability to do a remote firmware update for all commonly updatable firmware components.

5.3.1 Module in-Band Firmware Update Blocking from Composed System User

Required:

For a more secure design, the Compute module must provide a mechanism to block in-band firmware (such as BIOS, ME, BMC, NIC, DIMM, SSD, Firmware and so on) updates from a composed system user, but may allow updates from an administrator.

5.3.2 Firmware Update Authentication

Required:

The BMC, Intel® Management Engine (Intel® ME), BIOS, onboard NIC, and other component firmware updates must have an authentication mechanism to verify the update image before the Compute/Storage Module runs the updated firmware.

5.3.3 Module Configuration Default Support

Recommended:

If recovered from an unknown state (power loss during configuration change or bad combination of configuration settings or flash corruption, and so on), the module is recommended to provide a mechanism to restore module default settings (similar to entering the BIOS set up the screen and invoking restore default menu).

5.4 Module Configuration Information

Note: To make better workload placement decisions and have glass box insights into your datacenter, module configuration information needs to be exposed. The requirements for each specific implementation determine whether the Module configuration information is stored on power-up or provided in response to a request from the PODM or an orchestrator.

5.4.1 Module Processor Information (Out-of-Band)

Required:

The processor information details must be available through an out-of-band interface. Refer to Table 3, *Intel® RSD specifications* for API information details.

5.4.2 Module Memory Information (Out-of-Band)

Required:

Memory information details must be available through an out-of-band interface. This would include if the memory has persistent capabilities, (NVDIMM). Refer to Table 3, Intel® RSD API specification± for details.

5.4.3 Module Storage Information (Out-of-Band)

Recommended:

Local storage information details recommended are to be available through an out-of-band interface. Refer to Table 3, Intel® RSD API Specifications for details.

5.4.4 Compute Module Remote OS Boot Support

Required:

The Compute Module in the RSD must provide a mechanism to select the OS boot path remotely. Compute Modules must support either an iPXE interface or an iSCSI interface for a remote OS boot. If the Compute Module supports doing a remote boot, the user must configure the boot method (either local boot or remote boot).

5.4.5 Compute Module iPXE Support

Recommended:

It is recommended that Compute Modules support iPXE for remote starting. If it supports iPXE, the Compute Module should provide a mechanism to configure the iPXE parameters remotely through PSME/BMC. iPXE is much more secure than the Preboot eXecution Environment (PXE).

5.4.6 Compute Module iSCSI Support

Recommended:

If the Compute Module supports an iSCSI interface, the Compute Module should provide a mechanism to configure iSCSI parameters remotely through PSME/BMC.

5.4.7 Compute Module OS Boot from Local Storage

Recommended:

If local storage is present on the Compute Module, the Module should have the ability to support an OS boot from local storage. The local storage on the Compute Module should consist of resources such as SSD/HDD, M.2, or Non-Volatile Dual In-line Memory Module (NVDIMM).

5.4.8 Module Security Support Information

Recommended:

The Compute/Storage Modules are recommended to support RTM. Security support information (such as TPM presence, Intel® Trusted Execution Technology (Intel® TXT), and Authenticated Code Module (ACM) support) for a Compute/Storage Module is recommended to be given to the BMC/PSME to communicate the Module's security-related support information to the PODM.

5.5 Reliability Availability and Serviceability (RAS) Support

For improved overall data center operations and to understand the health of your datacenter at all times, health information needs to be exposed properly to the orchestrator. This section covers those requirements.

5.5.1 Out-of-Band Health Event Support

Required:

The BMC on a Compute/Storage Module must monitor hardware errors and health information for the Module (out-of-band, not dependent on the OS that is running on the Module), and provide the information to the PSME.

One mechanism for the PSME to receive health information from the compute module is to use the Intel® Intelligent Power Node Manager.

Refer to the *Intel® Intelligent Power Node Manager 4.0 External Interface Specification* as referenced in Table 3. Implementation must align with the APIs defined in the Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) Representational State Transfer (REST) API Specification Software v2.4.

5.5.2 Error Persistence Over Reboot

Recommended:

Hardware error information is recommended to persist across a reboot unless cleared by an administrator, or hardware is replaced or is otherwise corrected. Refer to Table 3, Intel® RSD API Specifications for details.

5.5.3 Reporting Health and Performance Information

Required:

The PSME must provide a mechanism to provide health and performance information to the PODM. Refer to Table 3, Intel® RSD API Specifications for error and health information reporting format details.

§

6.0 PCIe* Direct Attached Pooled I/O Design Guidelines

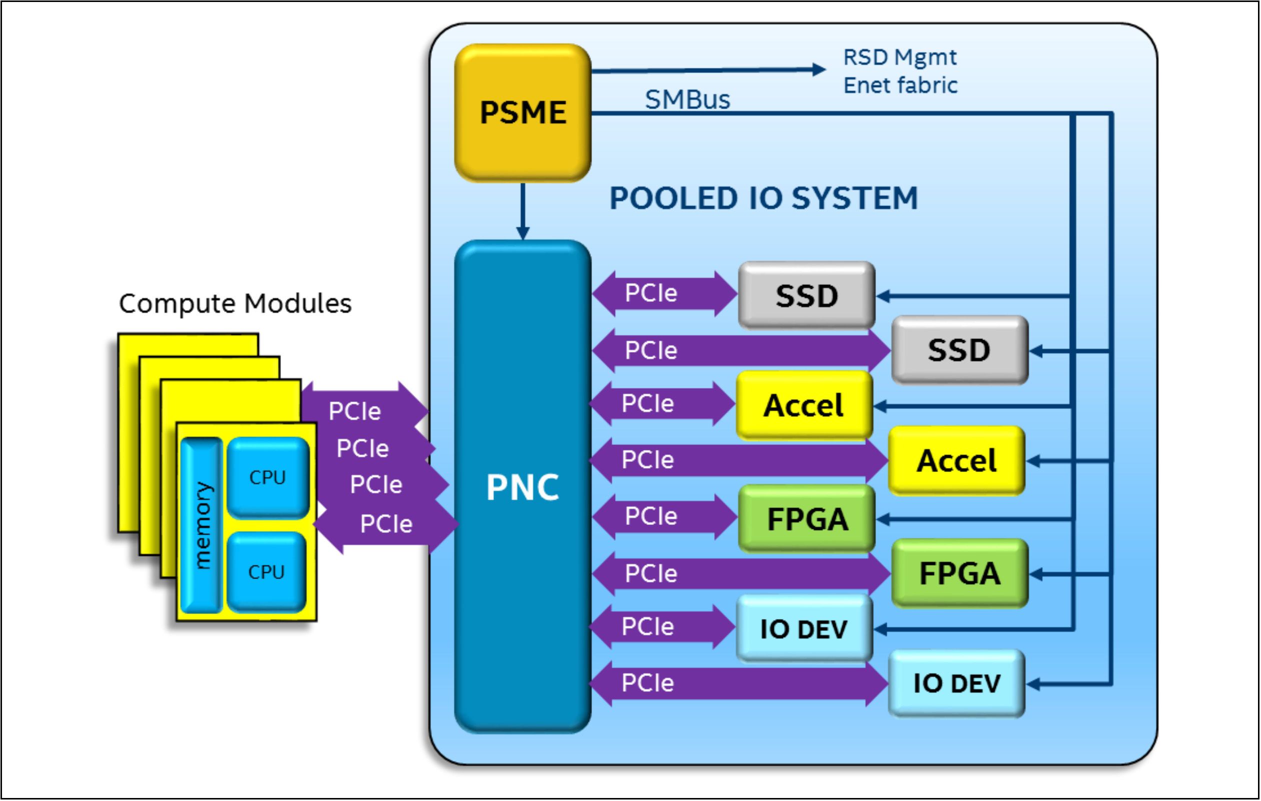
This section covers the requirements to support the PCIe Direct Attached Pooled I/O System, which is optional. This section also outlines the PCIe Direct Attached Pooled I/O related features and requirements containing PCIe I/O devices.

The PCIe Direct Attached Pooled, I/O System, provides disaggregation of I/O devices from the Compute Module. The I/O devices include storage and FPGA devices that are directly attached via a compliant PCIe bus interface.

Note: The optical transports can be used in connecting the I/O pool system, but they must be transparent to the PCIe interface at both the initiator and target ends.

The Direct Attached Pooled, I/O System, enables multiple nodes to access a pool of PCIe I/O devices providing flexible resource assignment and to maximize the efficiency and utilization of I/O resources. Within an Intel® RSD system, the PSME configures the I/O devices, Pooled Node Controller (PNC), and node connections to compose and release I/O resources within the I/O pool as instructed by the PODM. The assignment of these I/O devices to a particular node are accomplished through the PNC, by logically binding the PCIe endpoint devices to upstream ports attached to the compute modules in the system. It is possible for a single compute module to be attached to two or more PNC upstream ports for multi-pathing capabilities. Figure 15 shows an example of a pooled I/O system within the Intel® RSD architecture.

Figure 15. Example of a Pooled I/O System with PNC and PSMe*



6.1.1 Pooled Node Controller (PNC)

The PNC is responsible for the electrical and logical PCIe connections of the Compute Modules to the I/O Devices. The PNC function can be accomplished with a managed PCIe compliant switching device that meets the requirements of this document. Multiple PNCs may be used in a Pooled I/O System to achieve the I/O fan-out and connection topology desired by the Pooled I/O System, refer to Section 6.2 for details on system topology. The PNC is configured and governed by the PSME which requires a management port into the PNC.

Note: The command interface of the PSME to the PNC is not within the scope of this document as it is vendor specific to the PCIe switching device being used.

Connections to the PNC are accomplished through the PCIe ports of the switching device. Compute Modules having the PCIe root port are connected to upstream ports of the PNC while I/O devices are connected to downstream ports. The width of the port will be a function of its electrical connection and logical configuration.

Note: Peer-to-peer connections are not within the scope of this document.

6.1.2 Pooled Programmable Device Support

Intel® RSD 2.4 supports pooled programmable devices such as FPGAs which allow for a very versatile and flexible I/O system. Three primary programmable use cases are envisioned with programmable devices:

Dedicated Function

In this use case, the function of the device is defined from its local firmware load upon initialization. The device is responsible for loading its functional blocks and presenting itself as a particular device type to the PSME management plane. The function remains persistent with the local firmware, and neither the PSME nor the Compute Module may change its loaded function after initialization. Functional changes are only permitted via firmware updates which may be performed by the PSME by Intel® RSD specifications listed in Table 3. In this case, it is expected that the device function is fully operational at initialization time.

PODM Managed Function

In this use case, the PODM is solely responsible for managing the functional blocks, (bit streams), for the device. The baseline operational state, (i.e., static or blue bit stream), may be loaded by its local firmware enabling its primary electrical interfaces such as PCIe, memory and I/O channels. The PODM management can change or reprogram this baseline functionality as well as add or remove other functional blocks through the OOB programming channels. The persistence of the function instance is under the control of the PODM. In other words, the functional instance may remain across multiple composed assignments with other systems. The Host Compute Module is not permitted to alter the function of the device.

Compute Managed Function

In this case, the Compute Module can load functional blocks to the device via the IB PCIe channel. The device is expected to be initialized to a baseline operational state where at a minimum the PCIe interface to the PNC is functional for in-band programming. The PSME is responsible for returning the device to its original baseline operational state upon release of the device from the composed node and purging any functional blocks loaded by the Compute Module.

In all programming use cases the following criteria for the programmable device must be met:

- The OOB interface must be functional after power on or system reset so the PSME can discover and identify the device. Refer to Section 6.3, I/O Device Discovery Support for details.
- A means for the PSME to clear or purge any user content from the device and its local resources.
- The PSME can determine what programming use cases are supported as defined above. This will determine what agents are allowed to program the functionality of the device.
- Firmware updates to the programmable device are handled by the Intel® RSD specification.

6.2 System Topology and Mapping

To facilitate the composition of systems, the PODM must have a comprehensive view of the system topology to make effective decisions on how to logically assemble the system. At a high level, the PODM needs to know the following key attributes:

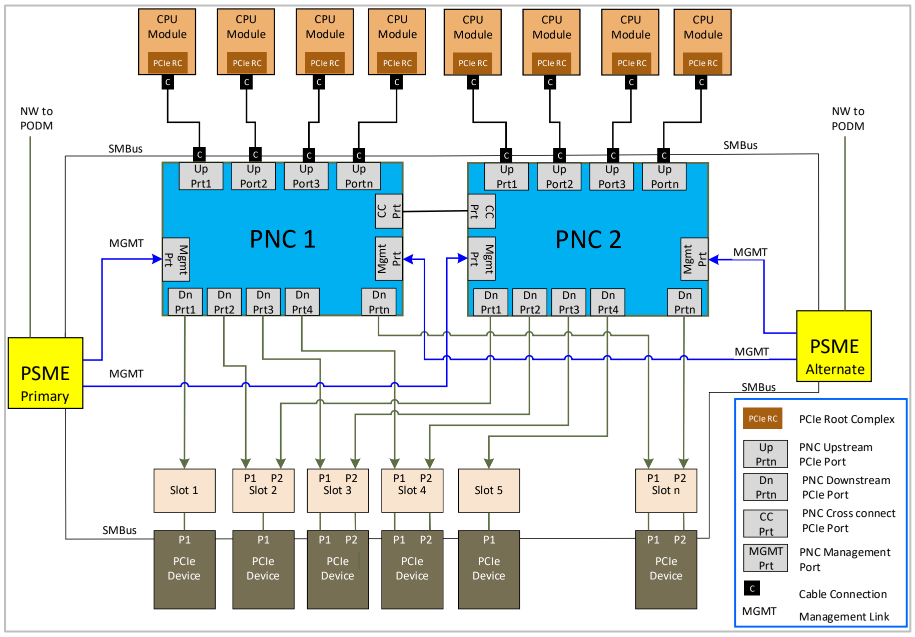
- The number of upstream ports that are available in the PNC(s), the widths of the ports, and the speed capabilities.
- Compute node attachment to the pooled I/O system; the PNC port number to which a compute node is attached.
- The number of downstream ports that are available, the widths of these ports, and the speed capabilities.
- How the downstream ports are physically mapped to the I/O devices or slots. It is possible that a slot will be wired to support more than one PCIe port (i.e., U.2 connectors may be configured for dual port devices).
- What devices are populated and the mapping to PNC downstream ports.

The PSME is responsible for exposing this information to the PODM through the Redfish-based APIs at the time of power-up or system reset. Once the topology is established, the PSME will discover the I/O devices, which are covered in Section 6.3, I/O Device Discovery Support. Discovery of the I/O device is accomplished through the out-of-band interface to the I/O devices described in Section 6.2.12, Expose the Connection Presence of each Upstream Port.

Multiple PSMEs may exist in a system, and they may be capable of managing one or more of the resident PNCs as shown in Figure 16. A Primary PSME will be designated in the case where multiple PSMEs have management access to a PNC. The Primary PSME will be responsible for the configuration and management of the PNC. Other PSMEs having management access will be designated as alternates and will have a passive or inactive role with the attached PNC.

The assignment of the Primary PSME may be predetermined by the I/O system but must be acknowledged by the PODM before configuring the I/O system. The PODM will be able to assign the Primary PSME if more than one PSME are present for an I/O pooled system. Refer to Section 6.2.5, Assignment of Primary PSME for PNC for more details.

Figure 16. Example of System Topology



6.2.1 Enumeration of Components in the System Must Be Deterministic and Persistent Across Power or Initialization Cycles

Required:

The enumeration of components, within the system, must be deterministic and consistent across power cycles or system initialization unless the system has undergone reconfiguration. This is required to provide the PODM a consistent view of the pooled system upon each power-up or system reset.

Such enumerable components include the PSME(s), PNC(s), external port connections, device slots, and downstream and upstream PNC ports.

6.2.2 PSME Exclusive Management Link to PNC

Required:

A private management link from the PSME processor to the PNC is required to configure and manage the PNC.

- The PSME shall support a separate management link to a Management Port for each PNC that the PSME is managing within its management domain.
- The link shall be Capable of mapping the device to the PSME Management link for firmware upgrade.
- The PNC shall be managed only via the Management port. Manageability includes configuration, telemetry, and firmware update.
- The link shall be Capable of exposing telemetry of the PNC device.
- Only one PSME may govern the PNC at any given time.

The PCIe switch that is used for the PNC in the system and is left to the designer for implementing the specific type, speed, and width of the management link. The management link is used to configure the PNC ports, establish any necessary PCIe domains within the PNC, collect telemetry information, and monitor any error conditions. The management link can also be directly mapped to an I/O device PCIe port for firmware upgrade and control.

6.2.3 Expose and Enumerate PNC Devices in a Pooled System

Required:

The PSME shall expose the total number of PNCs that are resident in the pooled I/O system. The PNCs will be logically enumerated so that the PODM can reference a specific PNC component. This will allow the PODM to logically map data connections from a compute node connected to an upstream port of an enumerated PNC to an I/O device.

6.2.4 Expose PSME Mapping of Management Connections to PNCs

Required if multiple PSMEs management connections are available to a PNC:

The PSME shall expose the mapping of its management interfaces to one or more enumerated PNCs resident in the pooled system. As shown in Figure 16, a system may have multiple PSMEs, each having a management link to one or more PNCs. The PSME shall expose management connection for enumerated PNCs.

6.2.5 Assignment of Primary PSME for PNC

Required if multiple PSMEs management connections are available to a PNC:

A Primary PSME will be assigned to a PNC:

- Only one Primary PSME may be assigned to a PNC at any given time.
- Only the Primary PSME may govern the PNC.

- The I/O pooled system may predetermine the assignment; it may be persistent across power or system reset cycles.
- All assignments must be acknowledged by the PODM before the PSME configuring the PNC.
- The PODM may assign the Primary PSME, overriding any I/O pool system assignment.
- Other PSMEs with management connections to the PNC will be assigned as an alternate.
- Alternate PSME(s) will take either a passive or an inactive role with the PNC. Any state information of the Primary PSME should be migrated to Alternate PSMEs.
- PODM notifies all PSMEs of their state of Primary or Alternate upon a change of state.

6.2.6 Expose and Enumerate PNC Upstream Ports

Required:

The PSME shall be able to expose and enumerate the upstream ports controlled by each PNC.

This information is used to establish connection paths of I/O resources on downstream ports to compute modules on the enumerated upstream ports.

Attributes to be reported:

- PNC number
- PNC upstream Port Number
- Port max lane width (optional)
- Port current lane width
- Port max speed (optional)
- Port current speed.

6.2.7 Expose and Enumerate PNC Downstream Ports

Required:

The PSME shall be able to expose and enumerate the PNC downstream ports configured within the PNC.

This information is used to establish connection paths of I/O resources on downstream ports.

Attributes to be reported:

- PNC number
- PNC downstream Port Number
- Port max lane width (optional)
- Port current lane width
- Port max speed (optional)
- Port current speed.

6.2.8 Expose Data Path Cross-Connections between Multiple PNCs

Recommended:

The PSME shall be able to expose the cross-connection topology between enumerated PNCs that allow devices to be logically distributed or assigned to compute modules attached to an adjacent PNC upstream port.

Generally, the cross-connection between PNCs provides a connection path from any of the I/O devices to a compute node port upstream. This provides a means for the PODM to determine the locality of the device to the compute node (such as the number of PNCs to reach the endpoint device) and to provision I/O Bandwidth (BW) appropriately across the cross-connection link.

Note: The PODM should first assign I/O resources to the host that does not use a cross-connection link.

6.2.9 Expose and Enumerate Device Slots of the I/O Pooled System

Required if the pooled system has connector slots for I/O devices:

The PSME shall be able to enumerate and expose the I/O slots and slot PCIe port configuration within the I/O pooled system. This will identify a PCIe slot, and port configuration of that slot within the I/O pooled system. This is independent of the slot being populated or unpopulated.

6.2.10 Expose Mapping of Device Slot Connectivity to PNC Downstream Ports

Required if the pooled system has connector slots for I/O devices:

The PSME shall be able to expose the mapping of slot ports to PNC downstream ports. As shown in Figure 16, device slots may be wired to have multiple PCIe port connections to one or more PNCs.

This will allow the PODM to logically map data connections from a PNC enumerated downstream port to an enumerated I/O device slot port. Consider the case where dual port SSDs connect to two independent PNCs providing the multipath capability.

Note: It is possible to connect a single port device into a dual port slot.

6.2.11 Compute Module to PNC Upstream Port Connection ID Mapping

Required:

There shall be a means for the PODM to map connections from a compute module to a PNC upstream port. Such a means may be accomplished by matching a pair of unique connection IDs from the compute module end and from the I/O pooled system end. In the case of removable cables, this may be accomplished by reading the cable ID field or serial number and matching it with the compute module end. Where connectivity is permanently set, such as in hardwired backplanes, a hard-coded connection ID could be presented to the PODM for mapping.

Refer to Section 6.8.5, Compute Module Connection Identification for information regarding connection identification for the Compute Module.

6.2.12 Expose the Connection Presence of each Upstream Port

Optional:

The PSME shall expose the state of the connection presence for each PNC upstream port. In the case of removable cables to make the connection to the compute module, this would be a cable present detect indication.

6.3 I/O Device Discovery Support

I/O Devices are discovered by the PSME and exposed to the PODM as I/O resources that can be composed and logically assembled with a compute module. For programmable devices such as FPGAs, the device must be programed or configured to an operational state that allows the PSME to discover it. Device discovery happens upon:

- Power up and initialization of the pooled system or subsystem
- Hot add of an I/O device

6.3.1 Expose the Presence of an I/O Device

Required:

The PSME shall have the means to expose the presence of an I/O device in a PCIe* slot. This is used to comprehend the population of I/O devices in PNC downstream slots.

6.3.2 Discovery of Device Type and Capability

Required:

The PSME shall be able to discover the I/O device, its type, that is, SSD or FPGA, along with capabilities such as capacity, performances, firmware, serial numbers, and any pertinent information for each device.

Mandatory:

- Type of device, that is, SSD, FPGA, and so forth
- Manufacturer ID, model, and serial number
- A number of PCIe* ports, width, and speed.
- For programmable devices what programmable use case is supported, refer to Section 6.1.2, Pooled Programmable Device Support

Optional:

- Storage capacity
- Performance
- Wear information
- Revision level of resident firmware local to the IO device
- Memory capacity
- Number of memory channels and speed
- Board service package rev level
- The function of programmable logic
- Other attributes or device capabilities.

6.3.3 PSME Configuration of I/O Device Support if Sharing of I/O Device Supported

Required:

The PSME shall have the ability to configure each I/O device individually if sharing of the I/O devices is supported. For example, if the NVMe* SSD is present, it should be able to partition the SSD before sharing the SSD between multiple compute nodes.

6.3.4 Expose Metrics for PCIe Devices

Recommended:

The PNC controller capable of providing:

- PCIe link metrics i.e. bus utilization, read and write activity
- SSD metrics per SSD device
- Programmable device metrics i.e. performance and throughput.

6.4 I/O Device Assignment to Compute Module

This section describes the requirements for logically assigning and releasing of device resources to compute nodes via upstream PNC ports.

6.4.1 Full Assignment of a Device PCIe* Function to a Single Compute Node

Required:

The PSME shall be able to connect the complete or full PCIe* function to an upstream port.

Note: If the system supports multiport or multifunction devices, each PCIe function can be logically assigned to the independent PNC upstream ports. As an example, a dual port device will present a PCIe function per port that can be assigned, or logically bound, to two different hosts or even to a single compute module using multiple upstream ports.

6.4.2 Assignment of Single PCIe* Function to Multiple Upstream Ports

Optional:

In Intel® RSD v2.1 assigning PCIe functions across multiple upstream ports is not supported to partition device resources. Virtual functions of a device may be assigned to the same PNC upstream port and compute module.

6.4.3 Dynamic Assignment of a Device Shall Not Affect Other Device Connectivity

Required:

Dynamic assignment of an I/O device ownership shall not affect other device-host connections.

This is accomplished via a hot-plug event to the system without the system having to be reset. This allows for the composition of I/O resources in logically assembling a system without disruption to other systems.

6.4.4 Dynamic Release of a Device Shall Not Affect Other Device Connectivity

Required:

The dynamic release of device ownership without affecting other devices-host connections. This is accomplished via a hot-plug event to the system without the system having to be reset. This allows for releasing I/O resources in disassembling a system.

6.4.5 Devices with Data Storage Must Secure Data Upon Release

Recommended:

Data stored on a device should be secured upon releasing the assignment of a device from the compute module. The data may be encrypted or fully erased to ensure the security of the data.

6.4.6 User Data Must Be Purged Upon Release of a Programmable Device

Required: If Programmable Devices are supported:

In all programmable device use cases, all user data must be purged from the device and any associated resources such as memory upon releasing the device from the compute module.

6.4.7 Functional Blocks Programmed by a Compute Module Must Be Purged Upon Release of a Programmable Device

Required if Compute Managed Function is allowed:

The PSME must have the means to purge any functional blocks programmed by a Compute Module and return the device to its baseline functional state upon the release of the device. Refer to the Compute Managed Function use case in Section 6.1.2, Pooled Programmable Device Support.

6.4.8 Persistence of Functional Blocks Instances Programmed by the PSME for Programmable Devices is under the Control of the PSME

Required if PSME Managed Function is allowed:

Functional blocks programmed by the PSME are under the control of the PSME. The persistence of the PSME programmed functions may remain with the device until deemed necessary to be removed by the PSME. This allows a programmed functional instance to remain in place across multiple composed assignments. Refer to the PSME Managed Function use case in Section 6.1.2.

6.4.9 I/O Resources Must Be Unassigned Before Assignment to a Compute Node

Required:

I/O device resources shall be in an unassigned or released state before they are permitted to be assigned to a compute node. Assigning an I/O device from one compute node to another compute node is not permitted without first releasing the I/O resource.

6.5 Adding or Removing Devices from the I/O Pool

Hot-plug is used for resource modifications for the pool such as drive addition or removal. The hot-plug event may be from a physical event, or it may be necessary to emulate the event by the management software on the PSME.

6.5.1 Physical Hot Add Support of Devices to the I/O Pool

Required:

A physical hot add of the devices to the I/O Pool shall be supported. The PSME will be responsible for the logical binding of the device to a compute node once present. Upon the addition of I/O devices, the PSME will be responsible for discovering the device. Refer to Section 6.3.

6.5.2 Managed Removal of Device from I/O Pool Support

Required:

The PODM shall notify all compute modules that are logically attached to the device to place them in a quiet state. The Compute Module will, in turn, notify the PODM when it has completed the task.

After all the Compute modules have reported their quiet state to the PODM, the PODM will inform the I/O Pool PSME the device is ready for removal. The PSME will then activate an indicator to notify personnel which device can be removed from the I/O pool.

6.5.3 Surprise Removal of a Device from I/O Pool Support

Required:

The compute module, and the PNC shall support surprise disconnect/removal of a device and continue in an operational state upon recovery of the disconnection. The affected PCIe link shall be functional upon adding a device to the slot. The PSME will alert the PODM of the event.

6.5.4 Surprise Disconnect of the I/O Pool Shall Be Supported

Required if removable connections to the pool are employed:

PCIe* busses connected to the Pooled I/O System that is removable shall support a surprising disconnect. The compute module shall be able to continue in an operational state upon recovery of the disconnection. The PSME will alert the PODM of this event.

6.5.5 Notification of Devices Added or Removed from the I/O Pool

Required:

Upon the physical addition or removal of a device, the PSME will alert the PODM of such an event. Upon addition, it will also report the device assets to the PODM.

6.6 Error Handling and Telemetry

This section describes the requirements for Error Handling and Telemetry.

6.6.1 Down Port Containment Support for All PNC Downstream Ports

Required:

All Downstream ports of the PNC shall support Down Port Containment. This will contain link failures in the event of a link down or surprise disconnect of a device.

6.6.2 Fault and Service Indicators for I/O Devices

Recommended:

The PSME is recommended to activate a fault or service indication for each device in the I/O pool. Typically, this would be an LED indicator located near the I/O device.

6.6.3 PNC Trap of PCIe Error Events Detected on the PCIe Link

Recommended:

PCIe errors are trapped within the PNC and sent to the PSME through the PNC management link. The PSME may log the error events or alert the PODM depending on the severity of the error and system policies. If the link is down and not recoverable, the PSME shall alert the PODM as a surprising disconnect.

6.6.4 Expose PNC, Device, and I/O Pooled System Telemetry

Recommended:

The PSME should be capable of reading telemetry information from the PNC, I/O devices, and enclosure, and be capable of exposing the information to the PODM. Refer to Section 7.3.4 for the details.

6.7 Pooled I/O System Support

This section describes the requirements for the pooled I/O system and chassis.

6.7.1 Device Serviceability while System Powered On

Required:

The Compute/Storage Modules must be able to service I/O devices without affecting running Compute/Storage Modules. To achieve this, the I/O chassis must support serviceability while the I/O chassis under power and must allow device serviceability without affecting other devices that are currently in use.

6.7.2 Pooled System Enclosure Management Support

Recommended:

Enclosure management of the Pooled I/O System and the I/O devices are provided through the PSME. This includes:

- Thermal monitoring within the strategic location of the chassis
- Thermal monitoring of each I/O device
- Voltage monitoring of each voltage rail
- Reset function to each I/O device in the I/O pool
- Reset function for the entire I/O pooled system
- Power on/off control
- Activate indicators for fault and attention for each I/O device
- I/O device power-on sequencing for power surge control if needed.

6.7.3 AUX Power to Cable Connector

Optional:

AUX power to the cable connection on both ends of the cable allows the PSME/BMC to read the cable information before power one of the system and establish connection mapping. Such fields for cable ID can be read and passed on to the PODM. The system mapping can be done before power on, saving time in the system coming up online, which should be done from both the Compute module and me/O pool endpoint of the cable.

6.7.4 Exposing Cable Electrical Parameters for Cable Signal Drive Support

Optional:

Reading cable loss characteristics and configuring the appropriate I/O drive capability of the cable electrical interface provides a means to optimize cable I/O drivers for various lengths and cable characteristics. This information may be stored and accessed via the I2C interface of certain cables. It is the responsibility of the local BMC at each end of the cable to perform this operation and done during power-up or detection of cable attachment. This should be done from both the Compute module and I/O pool endpoint of the cable.

6.8 Compute Module Requirements for IO Pooled Systems

This section describes the requirements necessary for the compute module to support a direct attached PCIe* pooled system.

6.8.1 Independent PCIe* Domain per Compute Module Connection

Required:

Establish independent PCIe domain per host, (zone), whereby actions or events in one domain do not affect the operation or state of other zones configured within the PCIe switch. Host issued reset only impacts its own zone (not the rest of the switch).

6.8.2 Down Port Containment Support for All Connected Ports

Required:

All Downstream ports of the Compute Module that connect to the I/O Pooled System shall support Down Port Containment. This will contain link failures in the event of a surprise link down or disconnect.

- Root must be configured to support completion time-out; this will create an all 1's completion for any pending transactions.
- The OS must be able to recover from a surprise disconnect; the PCIe driver must properly handle all 1's completion of the PCIe Transaction.
- Be able to reconnect to the device and establish the link upon reconnection of the link.
- PCIe Port that supports DPC should not set the Hot-Plug Surprise bit in the Slot Capabilities register; refer to the implementation note regarding surprise Hot-Plug with DPC in the *PCI Express Base Specification*, refer to Table 3.

6.8.3 BIOS Shall Allocate Memory Space for All Potential I/O Devices

Required:

BIOS must allocate memory space for all potential PNC slots for dynamic assignment of I/O devices.

1 MB, is the minimum allowed memory space defined by the *PCI Express Base Specification*, refer to Table 3.

6.8.4 Compute Module Visibility of I/O Device Controlled by the PSME

Required:

Host visibility to devices behind a switch port is strictly under the control of the PSME. Only when a device is assigned to a host will the host be able to enumerate and see the device. Compute modules will be inhibited to seeing any devices prior to the PNC being configured and enabled by the PSME.

6.8.5 Compute Module Connection Identification

Recommended:

Connection identification is supported on the compute module PSME.

The compute module presents a unique connection ID that corresponds to the I/O pooled system PNC upstream port connection. This would allow the PODM to determine the mapping based on matching connection ID fields from the compute module and PNC port of the I/O pool.

For removable cables, this could be the cable ID or serial number that matches the other end of the cable attaching to an I/O pool. For permanent connections, such as hardwired backplanes, this could be a hard-coded ID value.

6.8.6 Compute Module Managing the Assigned I/O Device

Optional:

Managing the assigned I/O device from the compute module is a platform design decision.

6.8.7 Compute Module Managing the Assigned I/O Device

Required:

Managing the assigned I/O device from the compute module is a platform design decision.

6.8.8 Compute Module Managing the I/O Pool System Is Not Allowed

Required:

Note: OOB management direct connection to the I/O pool from the host node is not permitted.

All OOB management is performed by the local PSME of the I/O pool. For instance, the Cable Management Interface, specified in the *PCIe External Cable Specification 3.0*, refer to Table 3, shall not be used as a management interface into the I/O pooled system.

§

7.0 PSME Design Guidelines

This section describes Intel® RSD Platform PSME design guidelines.

7.1 PSME Overview

The PSME is responsible for Drawer identification management, as well as supporting the PSME REST API and communicating with the BMC to perform Module-level management. In some implementations, RMM and PSME may co-exist in the same hardware. In some implementations and storage bricks, the PSME functionality may be provided by the BMC. In general, it is required to implement all of the required APIs as defined in Table 3, Intel® RSD Specifications.

7.2 PSME Reset (Power On)

Required:

Must be able to restart PSME services to troubleshoot issues remotely, and for overall improved data center operations.

7.3 PSME Configuration Management

This section describes the requirements for PSME Configuration Management.

7.3.1 PSME API Compliance

Required:

If the PSME is used, all required APIs must be implemented as defined in *Intel® RSD PSME REST API Specification Software v2.4*, refer to Table 3.

7.3.2 PSME Authentication Credential

Required:

For the PSME to identify and authenticate a PODM, the PSME must be provisioned with an authentication credential (***not an authorization credential***) that the PSME can use to verify the identity of the PODM.

7.3.3 PSME Time Sync Mechanism

Required:

For ease of troubleshooting overall improved datacenter operations, the PSME must log all management events with a timestamp. The PSME must synchronize time from the PODM periodically and keep the time tolerance within one second as compared to PODM.

7.3.4 PSME Telemetry Requirements

Required:

For ease of troubleshooting overall improved datacenter operations, the PSME must provide a set of telemetry data provided by the *Intel® RSD PSME REST API Specification Software v2.4*, refer to Table 3. It is recommended that all telemetry data be time stamped.

7.3.5 Serial-over-LAN and KVM Credential Change with User Changes

Recommended:

The PODM is recommended to change the credential for serial-over-LAN and KVM when the composed system user is changed. This is to ensure the previous composed system user does not have access to the same system. If the composed system is part of a user group, it may keep the credential the same.

7.3.6 PSME Support for Power and Thermal Capping

Recommended:

It is recommended that the PSME collect the power budget and capping capabilities, and provide them to the RMM (if discrete) or through the Intel® RSD REST APIs. The PSME can provide the monitored power consumption information for each Compute/Storage Module (and total of PSME-managed resource power consumption). In addition, the PSME is responsible for managing the fans in the Rack or Drawer, if present. Implementation must align to Intel® RSD API specifications listed in Table 3.

7.4 PSME Software Update

This section describes requirements for PSME software updates if a PSME is implemented in the solution.

7.4.1 PSME Remote Software Update

Required:

The PSME must be designed to accept online software updates or offline (staged) software updates.

For an online software update, the Drawer/Module functionality must not be lost during the software update process. However, during the PSME software update/reset window, the PSME may not respond. The PSME must inform its clients (such as PODM or RMM) before it goes through an online update process and must request the clients to reregister after a pre-specified length of time.

For an offline software update, the Drawers associated with the PSME are reset and during this time Drawer, functionality is lost.

Note: This mechanism must normally be performed during scheduled maintenance downtime, and the workloads must be shut down or migrated to a different Drawer before performing the update.

7.5 PSME Reliability, Availability and Serviceability Support

For improved overall data center operations, understand the health of your datacenter at all times, and to take appropriate action, the PSME is responsible for handling errors related to the PSME-managed assets, such as Drawers and Modules.

7.5.1 Drawer Event Reporting

Required:

For improved overall data operations and the ability to keep the datacenter running at all times, Drawer serviceability is required. To do that, the RSD APIs must support event reporting for Drawer insertion and Drawer removal events.

Drawer insertion:

- Once a Drawer is inserted and powered on the RMM must assign the PSME ID, and the RMM must advertise the new PSME to the PODM.
- The PSME must provide the Drawer details such as Compute/Storage Module presence information to the PODM.

Drawer removal:

- If the PSME is not responding, it is either due to PSME failure or due to Drawer removal. If the RMM is managing the drawer it must periodically poll it for presence to manage power and cooling. The PODM shall also poll the PSME for availability in managing the PSME.
  - Module removal
  - Module failure events, such as boot failure detected by BMC watchdog timer

Implementation must align Intel® RSD API specifications listed in Table 3.

7.5.2 Drawer (PSME) Hot Add Only when RMM Is Present and Running

Recommended:

To avoid periodic discovery of the system components, it is recommended to insert a Drawer only into a Rack that has an active RMM, if RMM is present. If the RMM fails, the PSME IDs are kept intact to keep the Composed Nodes running.

This avoids the situation where a Drawer is moved into a Rack with an RMM that has failed, and the PSME continues to provide access to Composed Node resources provided by that Drawer.

Furthermore, it is recommended that the PODM detect RMM failure. If any new PSME shows up in a Rack with an inactive RMM, it is recommended that the PODM reject the new PSME until the RMM is active again.

§

8.0 RMM Design Guidelines

This section describes the Intel® RSD Platform RMM design guidelines. The RMM API is a required component for RSD, but the hardware that runs the RMM firmware could be dedicated or shared with PSME or other components. Either the RMM can be ran as a separate component, or APIs can be implemented in the PSME or PODM (for example).

8.1 RMM Overview

The RMM is responsible for handling infrastructure functions such as rack level shared power; rack level shared cooling, and assigning PSME IDs.

8.2 RMM Reset (Power On)

Required:

The RMM must be configured first with PODM authentication credentials (Refer to Section 7.2.2 for details). Once the credentials are configured, the RMM can communicate with the PODM without any external configuration.

8.2.1 RMM Boot and PSME ID Assignment if Discrete RMM Present

Required:

If a separate RMM component is present, the RMM must assign a unique PSME ID for each PSME managed by an RMM.

Note: The conditions listed in Table 8 must be met during the RMM boot process and the PSME ID assignment processes.

If redundant RMMs are configured for high availability support in a Hyper Scale datacenter, the existing Composed Nodes can continue to operate in the event of an RMM failure.

Table 8. RMM and PSME Interaction during Boot and ID Assignment

RMM Condition	PSME Condition	Requirement Specification
During runtime, primary and secondary RMMs running and primary RMM fails	During runtime, previously configured PSME running	The secondary RMM must take control from primary RMM without changing the PSME ID.
During start, RMM failed or no RMM found	During start, previously configured PSME resource found and running	PODM has the list of PSMEs that was previously running, reports the RMM failure, and continues to use the PSME.
During start, RMM failed or no RMM found	New PSME (Drawer) is added to the Rack. Since PSME is not finding RMM, the PSME ID won't be set	The PSME must wait for the RMM to assign the PSME ID and complete the boot operation. The PODM will not use the new PSME.
During start, RMM finds no PSME	No PSME ID will be allocated	PODM won't use PSME
During start, RMM present	During start, PSME present	The PSME must ping the RMM by sending the active PSME ID and continue to boot until done.

8.2.2 RMM Assigns PSME ID if PSME Not Configured

Required:

If a separate RMM component exists, and if the RMM finds a PSME that reports itself as not being configured, then the RMM must configure the PSME ID.

This condition occurs under two conditions: when the RMM is reset, and when a PSME is hot added.

8.2.3 PSME Enters "PSME ID Not Configured" State

Required:

To properly discover, manage, and recover from newly added Intel® RSD components after a failure scenario, if a new RMM is found during a PSME boot process (as a new insertion to rack or for a redundant configuration for example), the PSME will not complete the boot process until it is configured with a PSME ID. If the same RMM is found, then the PSME must retain the old PSME ID until the RMM assigns a new ID.

8.3 RMM General Support

8.3.1 RMM Event Handling

Required:

Similar to PSME, the RMM, if present, must handle the following events and report the events to the PODM:

- Drawer Insertion:
  - New PSME detected and the PSME is assigned with an ID
  - New drawer location identified
- Drawer Removal:
  - PSME removed from the RMM list
  - Drawer removed location identified
- RMM internal errors reported
- (If High Availability (HA) RMMs are present), a new RMM becoming the active primary after a redundancy loss is signaled
- Power events such as a new power supply coming online or power supply failure
- Power threshold crossing events
- Cooling threshold crossing events

All events must align to the *Intel® Rack Scale Design (Intel® RSD) Rack Management Module (RMM) Representational State Transfer (REST) API Specification* definitions. Refer to Table 3.

8.4 RMM Power and Cooling Support

8.4.1 Rack Power Monitoring Support by RMM if Shared Power Is Used

Required:

If rack-level shared power is used, the RMM must provide power-monitoring support for power supplied to the rack. Refer to Intel® RSD API specifications; refer to Table 3, for the data format.

8.4.2 Rack Power Budgeting Support by RMM if Shared Power is Used

Recommended:

If the rack level shared power is used, it is recommended the RMM provide setting a power limit for racks. Refer to Intel® RSD API Specifications, Table 3, for the data format.

§

9.0 POD Manager (PODM) Design Guidelines

This section describes the PODM design guidelines used on the Intel® RSD Platform.

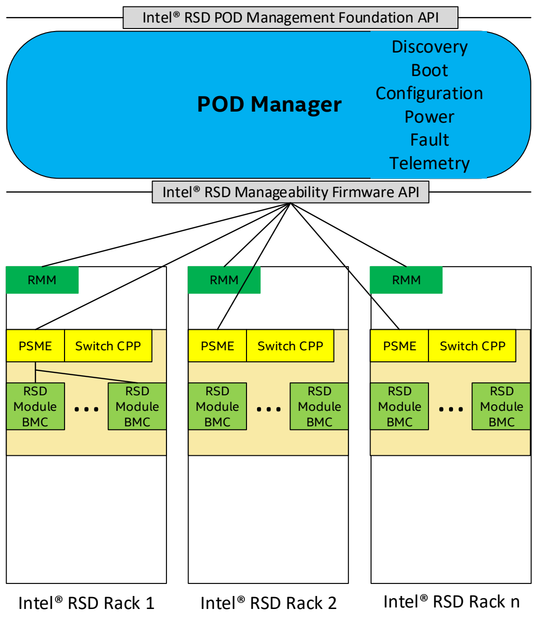
9.1 PODM Overview

The PODM, shown in Figure 17, is responsible for the discovery of resources in the POD, configuring and managing the resources, and composing a logical server.

The PODM is an optional separate component and will often not be required in-rack. However to be 'RSD conformant' a Rack must be able to be managed by a certified PODM. This section outlines requirements for those who implement a PODM solution. All APIs, as defined in the *Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) Representational State Transfer (REST) API Specification*, must be implemented as defined in that spec. Refer to Table 3.

When working within the POD, the PODM interacts with RMMs in the Rack and PSMEs to create a representation of the Intel® RSD POD. The PODM assigns the physical resources to match the requirements specified by the Intel® RSD Solution Stack by creating a logical resource, called a composed node or logical server.

Figure 17. Logical View of the Intel® RSD POD Manager



9.2 PODM Configuration Management

9.2.1 PODM Powered Independent of Rack Power

Required:

To maintain the PODM services even when the rack is reset, the PODM must be independently powered.

9.2.2 PODM REST API Compliance

Required:

The PODM must be in compliance with the *Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) Representational State Transfer (REST) API Specification*. Refer to Table 3.

9.2.3 Secure Communication Channel for Management Network

Required:

For a more secure design, the PODM must be connected to the RMM (if present) and PSME through a private network. Any management-related activity (such as reconfiguration) must be performed only after establishing a secure communication channel between the PODM and the PSME, and between the PODM and the RMM.

9.2.4 PODM Authentication Certificate

Required:

For a more secure design, the PODM must securely identify or authenticate itself to the PSME and RMM (if present) using a PODM authentication certificate. One way to do this (and there are others), is for the PODM to create a private/public keypair, then request a certificate (from a certificate authority) that includes a public key for this keypair. The certificate is provisioned to RMMs and PSMEs that use it to authenticate this PODM (Refer to Section 7.3.2, PSME Authentication Credential for more details).

9.2.5 PODM Timestamp Support

Required:

For better troubleshooting and improved overall data center operations, the Intel® RSD components must log all events with a timestamp. One-way to achieve this is for the PODM to synchronize the time using syslog or NTP.

9.2.6 Multiple or Distributed PODM per POD Access Considerations

Required if multiple or distributed PODMs are managing a POD:

Note: If multiple PODMs are able to manage a POD then the following requirements must be met.

- Each PODM must use the same credentials in accessing the RMM and PSME within a POD.
- The RMM and PSME configuration must be an autonomous PODM operation.

9.2.7 PODM To Allow Addition of New Drawers Only When RMM Is Alive

Required if the RMM is managing the Drawer:

If a new Drawer and new PSME) are added to a rack, the PSME is not activated with PODM credentials without the RMM in an active state to be able to evaluate the power and thermal load to the rack.

The intent is to prevent the addition of a new Drawer to unduly take down or throttle other systems in the rack for lack of available power or cooling resources.. For this reason, the PODM *must not* be able to use this new drawer for composing nodes if the RMM is not active. The PODM shall poll the RMM to determine its state prior to activating the PSME.

§

10.0 Network Switch Design Guidelines

This section describes the design guidelines for the network switch configuration.

10.1 Intel® RSD Networking Overview

In a typical Intel® RSD Platform, Compute/Storage Modules are connected at the Module level to a NIC. For switch management, the PSME could be on the switch itself or could be external to the switch.

In Intel® RSD versions before v2.2, the PSME network agent is designed for managing Red Rock Canyon (RRC). RRC switches do not provide a network-accessible configuration command set, the RSD provides APIs, and an implementation for configuring RRC switches; the same APIs have been implemented by OEMs for configuration of TORS.

Starting with Intel® RSD v2.2, RSD moves to Top-of-Rack switches, which support comprehensive configuration command sets. RSD's goal is to enable network vendors to align with a common configuration model and an evolving industry standard. Until this goal is available, OEMs and users are encouraged to leverage industry recognized and widely adopted configuration management tools (for example, Ansible*) which provide scalable configuration of a switch from numerous vendors.

Note: The switch APIs from previous versions are still defined in the PSME (and PODM) API specifications in Table 3, but will be removed in a future Intel® RSD revision.

10.1.1 Module-to-Port Mapping Configuration File Support if Dynamic Discovery Not Supported

Required:

If the dynamic discovery of the mapping between switch ports and Modules is not possible, then there must be a configuration file that is available in the PSME to describe the physical connections between the switch ports and the Modules for the Intel® RSD management software.

10.1.2 Switch PSME Support for Base Network Services

Required:

The switch PSME must expose the interface/API for the following base network services:

- HW management
- Interface Management - user can configure switch port interfaces
- Protocol Management
  - VLAN configuration
  - IDs for switch port neighbor ports (for example, mapping switch ports to NIC port MAC address)
  - MAC Address - view and configure switch MAC address tables for defining rules about which packets are forwarded or discarded.

10.1.3 Device Discovery and Switch Configuration Reporting

Required:

The PSME switch should provide initial device discovery and switch system configurations before computer systems can be composed; this is performed autonomously without initiating a request from PODM.

10.1.4 Switch Functionality Change Event Generation

Required:

If any of the following conditions occur, the switch PSME should generate an event to PODM and notify the switch state:

- Port is not functional
- Port state is up or down
- Link state is up or down.

§

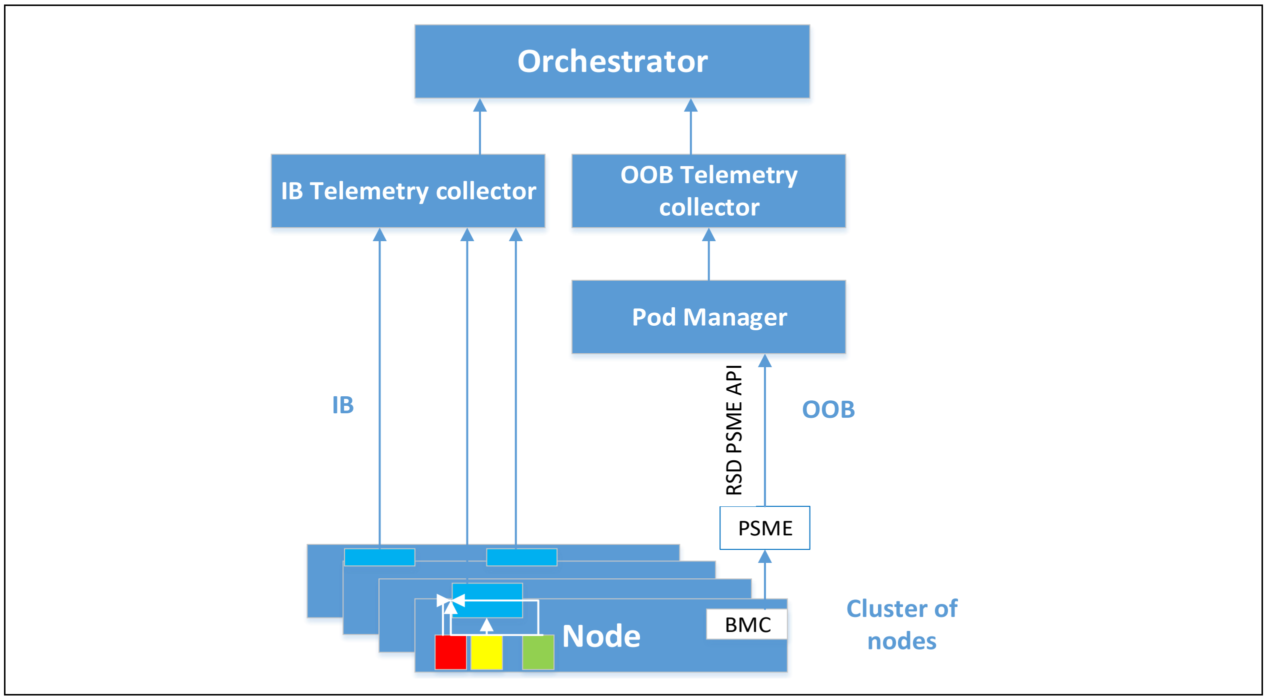
11.0 Telemetry

The Intel® RSD platform contains various compute, storage, and communication elements with varying power and performance characteristics. By monitoring various parameters in these elements and taking appropriate actions, optimal performance and lower TCO can be achieved.

11.1 Intel® RSD Telemetry Architecture Overview

Intel® RSD Telemetry Architecture allows telemetry collection through an OOB interface, through the *Intel® RSD PSME REST API Specification*, Table 3, and through an in-band (IB) interface as shown in Figure 18.

Figure 18. Intel® RSD Telemetry Architecture



RSD exposure of telemetry is classified under the following categories:

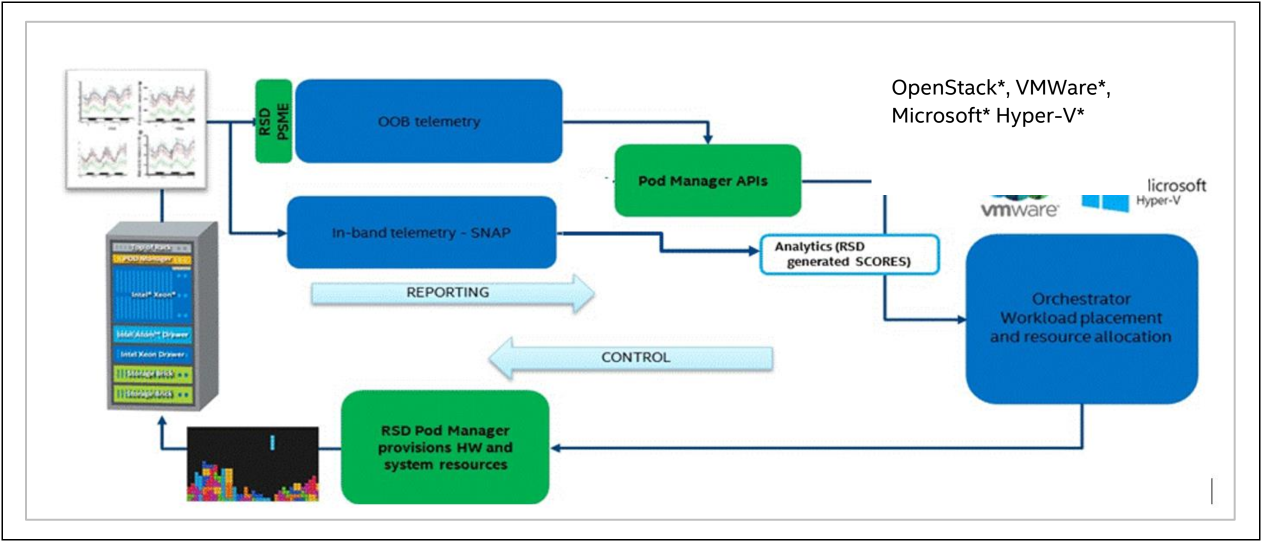
- State
- Event
- Configuration
- Available Capacity
- Performance

Refer to Section 11.3.1, PSME API Support for Telemetry for specific telemetry features supported for each RSD revision.

11.2 Monitoring Architecture for Intel® RSD

Figure 19 shows a typical Intel® RSD telemetry flow, where the telemetry data is collected from compute, storage, and network devices. Telemetry is provided to the PODM through the Intel® RSD PSME API interface for OOB telemetry items and through an in-band interface for the items that are not generally available through the OOB interface.

Figure 19. Typical Telemetry Flow



11.3 Telemetry Support Requirements for Intel® RSD

Needs Intro

11.3.1 PSME API Support for Telemetry

Required:

The PSME API must support telemetry features listed in Table 9. The PSME communicates with BMC/ME/BIOS to provide the following functionality. Refer to the *Intel® RSD PSME REST API Specification*, Table 3, for supported telemetry metrics.

Table 9. Intel® RSD Telemetry Support Summary

Component	Sensor/Usage	Required/Recommended	Support Starting Intel® RSD version
Processor	Health	Required	2.2
	Utilization	Required	2.2
	Average Frequency	Required	2.2
	Throttling	Required	2.2
	Temperature	Required	2.2
	Consumed Power	Required	2.2
Memory	Health	Required	2.2
	Bandwidth	Required	2.2
	Throttling	Required	2.2
	Temperature	Required	2.2
	Consumed Power	Required	2.2

Component	Sensor/Usage	Required/Recommended	Support Starting Intel® RSD version
Chassis (SLED) metrics	Inlet Temperature	Recommended	2.2
	Outlet Temperature	Recommended	2.2
Computer System	IO Bandwidth	Required	2.2
PNC	Port Health	Required	2.2
Pooled NVMe drives	Health	Required	2.2

11.3.2 PODM SB and NB API Support for Telemetry

Required:

For the orchestration software to take advantage of the telemetry data, the PODM North Bound (NB) API must support the telemetry APIs. In turn, the PODM must implement the South Bound (SB) telemetry API to get the data from the PSME.

11.3.3 Support for In-Band Telemetry Collection

Recommended:

Not all telemetry data are available through the OOB interface. It is recommended the Intel® RSD platforms support the in-band telemetry data collection service for the orchestration to take full advantage of the telemetry data.

11.3.4 Support for Correlation of IB and OOB Telemetry Data

Recommended:

Incorporate IB and OOB metrics and event data collection at the “Telemetry Service”, which can be hosted with the PODM service and correlate the in-band data with the OOB data at the RSD controller layer.

§

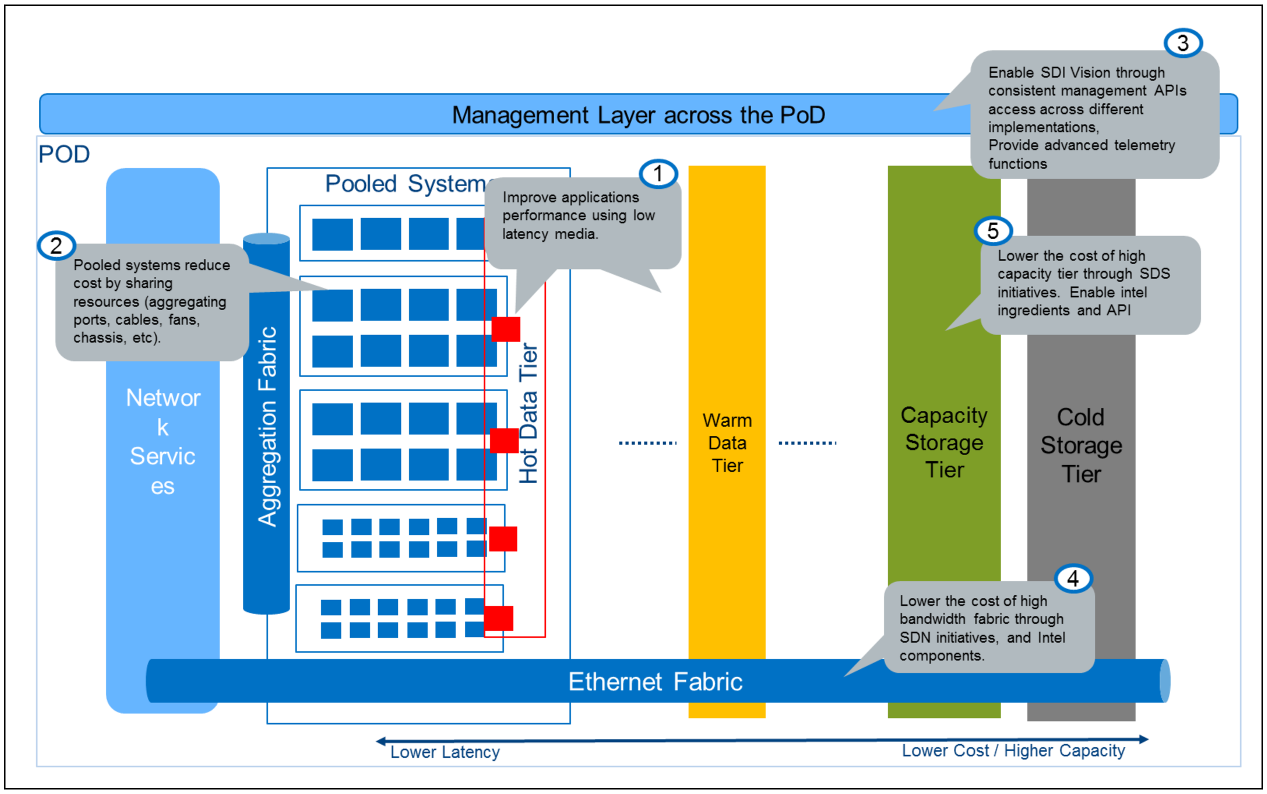
12.0 Ethernet Pooled Storage

12.1 Overview

One of the goals of Rack Scale Design is to disaggregate Storage and Compute elements of a rack to realize the Intel® RSD 2.4 vision outlined in Figure 20. The hot and warm data tier can be provided using PCIe and/or high-speed fabric attached storage. A solution, code named Sellwood Bridge, delivers the pooled PCIe attached storage element of this vision. The fabric attached storage element will be delivered using NVMe-oF storage solutions. Specifically this document will cover the functional and architectural considerations for delivering NVMe over RDMA/Ethernet as part of Rack Scale Design solutions.

Note: Not all the required, recommended, and optional items are embedded into the text and not fully reflected in Table 4.

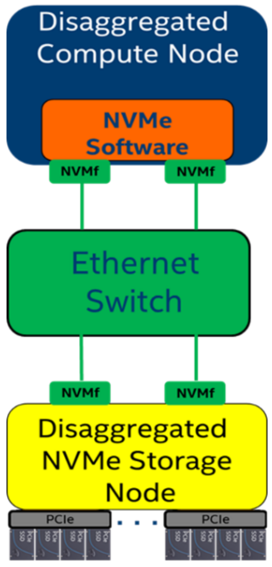
Figure 20. Intel® RSD Vision


Management Layer across the PoD
POD
Pooled System
Network Services
Aggregation Fabric
Hot Data Tier
Warm Data Tier
Capacity Storage Tier
Cold Storage Tier
Ethernet Fabric
Lower Latency
Lower Cost / Higher Capacity
1 Improve applications performance using low latency media.
2 Pooled systems reduce cost by sharing resources (aggregating ports, cables, fans, chassis, etc).
3 Enable SDI Vision through consistent management APIs access across different implementations, Provide advanced telemetry functions
4 Lower the cost of high bandwidth fabric through SDN initiatives, and Intel components.
5 Lower the cost of high capacity tier through SDS initiatives. Enable intel ingredients and API

12.2 NVMe* over Fabric Topologies

The NVM Express Organization has defined an architecture that layers a fabric protocol over the NVMe* host access mechanism to extend the NVMe interface across a network to remote hosts as shown in Figure 21.

Figure 21. NVMe-oF Basic Topology

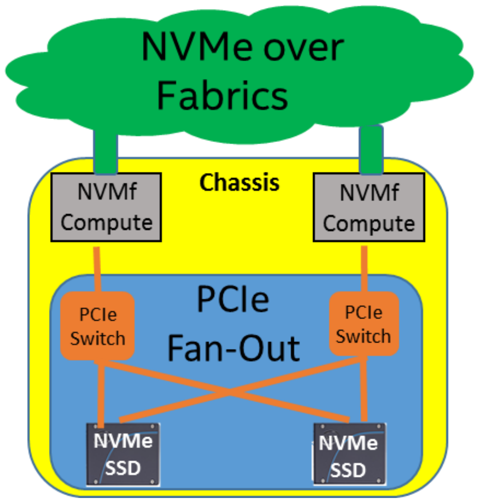


The current specification focuses on Ethernet as the fabric that uses RDMA as the transport protocol for the NVMe commands between the compute and storage nodes.

The disaggregated compute node is known as the `“host”` (similar to an `“Initiator”` in networked storage terminology). The disaggregated storage Target enclosure may contain several `“NVMe Subsystems”` (similar to a Target in networked storage terminology). Two types of target subsystem topologies are expected in solutions.

12.2.1 IA Processor-Based

Figure 22. NVMe-oF Processor-Based Subsystem

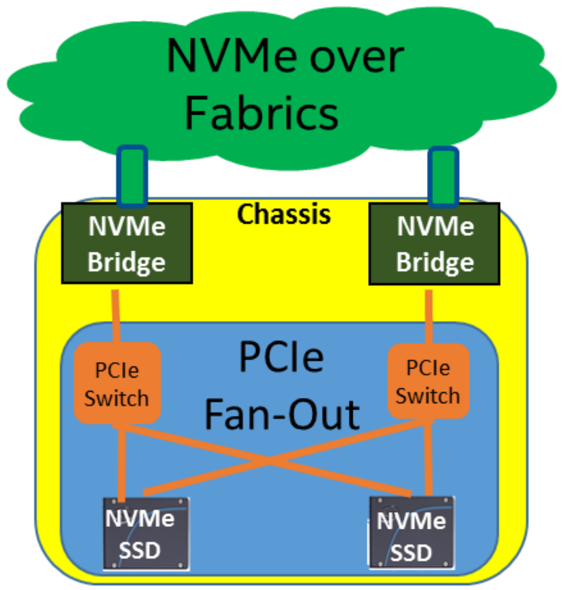


In this model, the translation between the NVMe* commands over the RDMA transport to the PCIe* NVMe SSDs is done in software running in the `"NVMe-oF Compute"` processors in the chassis. The chassis will have RDMA-capable NICs and a PCIe complex that houses the SSDs. Typically the software will provide other advanced storage services such as storage virtualization, data protection across physical disks inside the chassis, replication across chassis, data compression, deduplication, and so forth. Both the reference NVMe-oF target implementation in the Linux* kernel and Intel's Storage Performance Development Kit (SPDK) follow this model.

If the target implements storage virtualization, then the NVMe controllers and their attached namespaces that are exposed to the remote host may not have any direct relationship with the physical NVMe drives in the system. These controllers and their storage will likely be volumes composed of parts of several physical drives.

12.2.2 Hardware NVMe-oF Bridge

Figure 23. HW-Based Subsystem



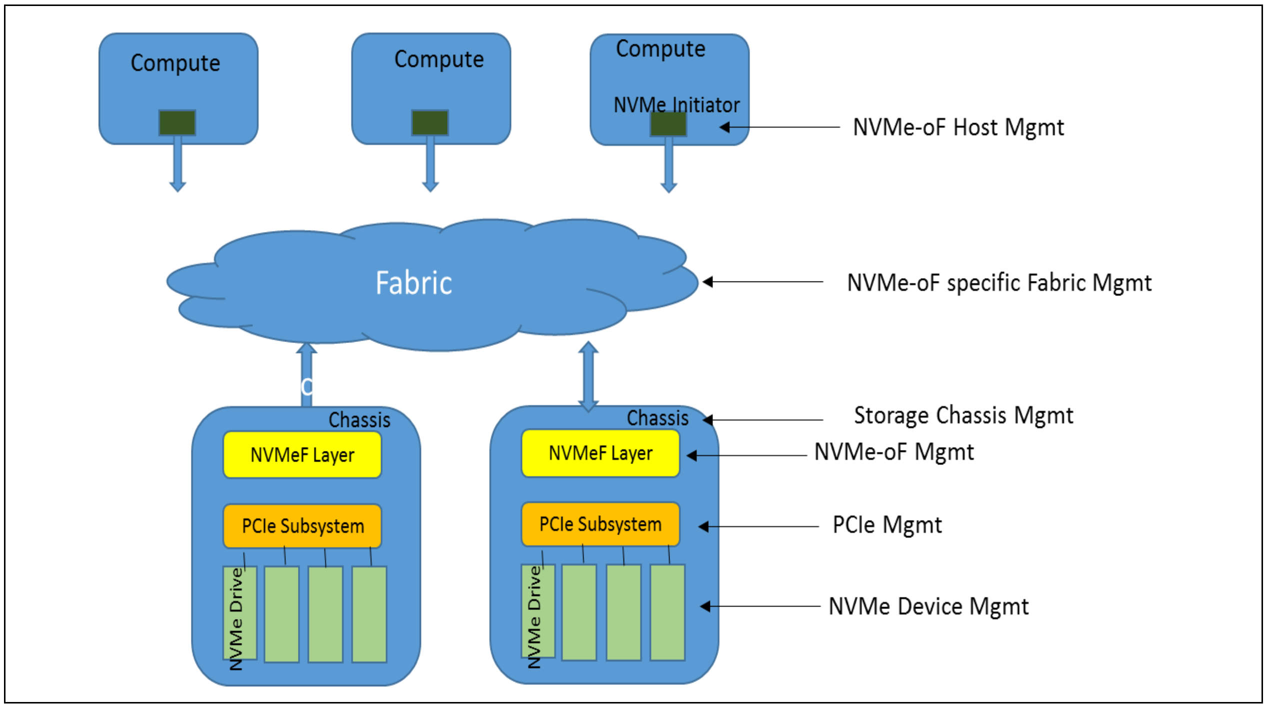
In this model, the translation between the RDMA transport and NVMe commands to the physical SSDs is done inside an FPGA or Application Specific Integrated Circuit (ASIC) (`"NVMe Bridge"` in Figure 23).

Generally, these hardware components have integrated Ethernet ports on one side and connect to a PCIe complex on the other housing the NVMe SSDs. While it is possible that these can deliver advanced storage functionality, the most likely implementation would be targeted towards a low-cost, high-performance solution that exposes the physical drives directly over the fabric as a bunch of Flash disks.

12.3 Management Model

To realize a comprehensive management of NVMe pooling over Ethernet, the management model spans across several domains.

Figure 24. Management Domains in an NVMe* over Fabrics Pooled Deployment



Regardless of the type of NVMe-oF target subsystem in use, there will likely be a storage enclosure or chassis, which houses the storage devices and processing elements. The physical chassis will be managed via the RSD Chassis management interface (part of Intel® RSD v1.2) as a Storage System.

The physical storage devices will be NVMe* controllers attached to the processing elements via a PCIe complex. These physical NVMe devices will be managed by its OOB management interface NVMe* Management Interface (NVMe-MI*), being defined by the NVM Express organization. This management interface will provide information about the drive such as serial numbers that will allow correlating that with the drive information provided via the NVMe-oF layer.

The PCIe complex may be either a static or a dynamic tree that supports pooling. If a PSME exists for the PCIe complex, then it can be managed via the RSD PCIe device management.

The NVMe-oF layer in the target subsystem provides all the NVMe-oF storage and protocol functionality. This specification will focus on the functional requirements of this layer in support of the Intel® RSD usage model.

The host side element of NVMe-oF is also a critical element in managing pooling because it participates in connecting the compute node to its pooled storage as part of a logical node composition. This specification will address the requirements of this layer to support the Intel® RSD usage model.

The network is a last key piece of the puzzle since it provides the physical connectivity that enables the NVMe-oF protocol to disaggregate NVMe storage. This specification will address the requirements of the fabric to enable efficient usage of a converged network for NVMe-oF pooling.

12.3.1 NVMe-oF Discovery and Mapping

The discovery and mapping of hosts to disks in an NVMe-oF deployment is done by overlaying a logical connection over the physical network topology. The logical connection between a host and a Subsystem (target) is done by the host initiating a transport-level connection with an NVMe subsystem using fabric-specific transport mechanisms (for example, RDMA send/recv, Queue Pair establishment) and then subsequently initiating a NVMe-oF level connection to the subsystem using the `NVMe-oF CONNECT` command.

The first step is for the host to discover all the NVMe* Subsystems in the network it has access too. This can be done via RSD (OOB) or via a Discovery Service provided by an NVMe Subsystem that supports only Discovery controllers. This controller has the same access mechanism as regular controllers except that its job is to provide the Discovery Log Page to a requester instead of namespaces and actual data services. The Discovery Log Page contains entries that specify information for the host to connect to an NVMe-oF Subsystem.

The subsystem specified in an entry may be one with namespaces or another discovery controller. The actual list of entries returned to a host may vary since the entries are specific to a host. The host uses a well-known NVMe Qualified Name (NQN; `nqn.2014-08.org.nvmexpress.discovery` in the *NVM Express over Fabrics Specification*; refer to Table 3) to connect to the first discovery controller.

Note: The resolution of the well-known NQN to an IP address (for Ethernet as the fabric) is not defined in the NVMe-oF specification and is host implementation dependent.

After the first Discovery Controller is connected, then the log entries from that controller will specify the fabric-specific address of the other controllers. If an RSD POD is introduced alongside a non-RSD POD, which already has an NVMe-oF discovery service, the RSD POD will have either to interface with the existing service, or provide its own only for the resources and hosts within the POD.

12.3.2 Logical Intel® RSD Management Model

As mentioned in previous sections, an NVMe-oF deployment requires the participation of the host (Initiator), network, subsystem (target), and a discovery subsystem. Therefore, it follows that the Intel® RSD management model requires a management interface and functionality for each of these elements as shown in Figure 25.

Each of the three NVMe-oF functional elements will require an RSD agent that talks to the PODM to allow discovery and configuration of NVMe-oF services. These agents are logically equivalent to the PSME/PSMF in the PCIe* Pooled storage architecture but may not have the same implementation. The interface to the PODM for these entities shall be an extension of the Swordfish*/Redfish* model, Figure 26, which shows two realizations of the NVME-oF target management entity based on the two common target implementations. As shown, the x86 full stack target implementation will have the Intel® RSD management agent running alongside the storage stack similar to the Intel® RSD storage services agent for iSCSI.

On the other hand, a HW bridge-based implementation will have a control processor that can host the target management agent similar to the PSME in the pooled PCIe storage implementation. This specification will be referring to these agents as NVME-oF Pooled Storage Management Service (PSMS) regardless of the actual realization.

The PSMS for an NVMe-oF target enclosure will be responsible for enumerating the virtual and physical data objects as well as data services provided by the target. It will also support configuration of said data objects and services within the constraints of the NVMe-oF subsystem implementation.

Similarly, the PSMS for the NVMe-oF Discovery controller is a specialized agent that enumerates the discovery services and allows configuration of the Discovery controller to provide automated set up of Host-Subsystem discovery/association.

The PSMS for the NVMe-oF host (Initiator) is responsible for reporting the existence and capabilities of the host vis-à-vis NVMe-oF and allows its configuration for discovery purposes. The host may implement a software or hardware-based initiator. For a hardware-based initiator the PSMS will be the BMC/PSME on the host and can report the existence of the initiator (and its capabilities). However, for an SW initiator (typically in the OS kernel), the PODM may configure it pre-OS but it is not required to ensure that the initiator is installed on the host.

The Fabric (or Ethernet Network) does not require an agent specifically for NVMe-oF, but may be required to be configured to allow specialized treatment of storage traffic with respect to priority, bandwidth, and congestion management.

Figure 25. Intel® RSD Management Entities for NVMe-oF Pooling

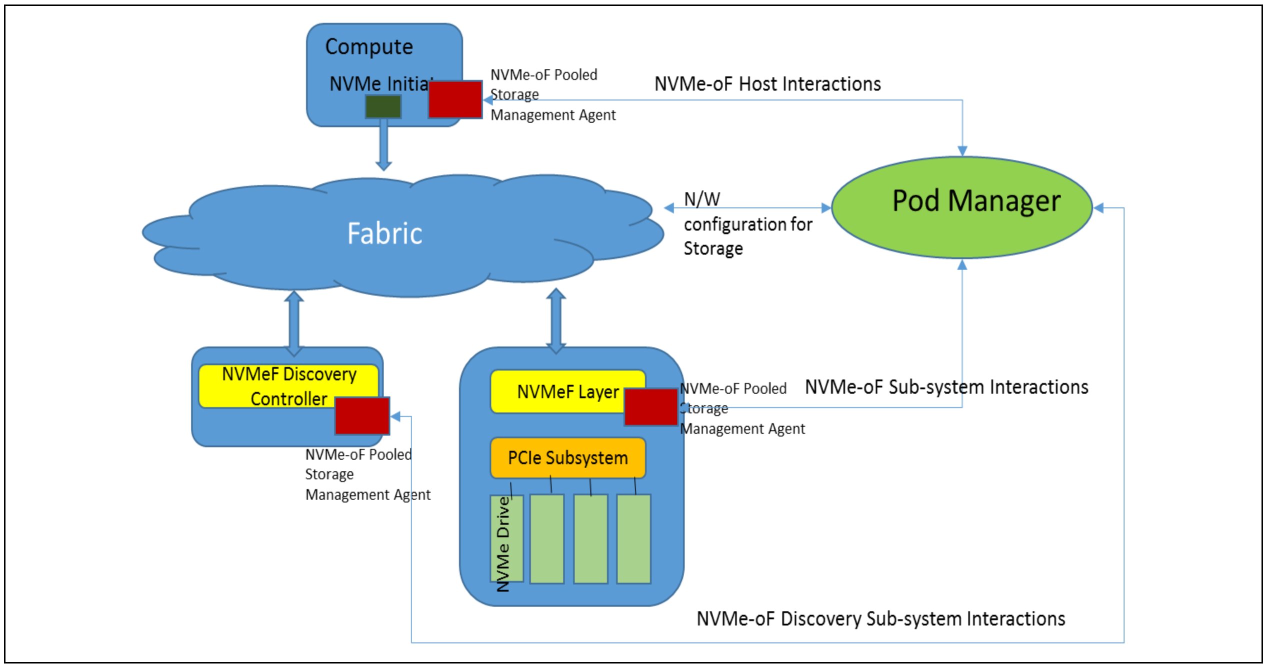
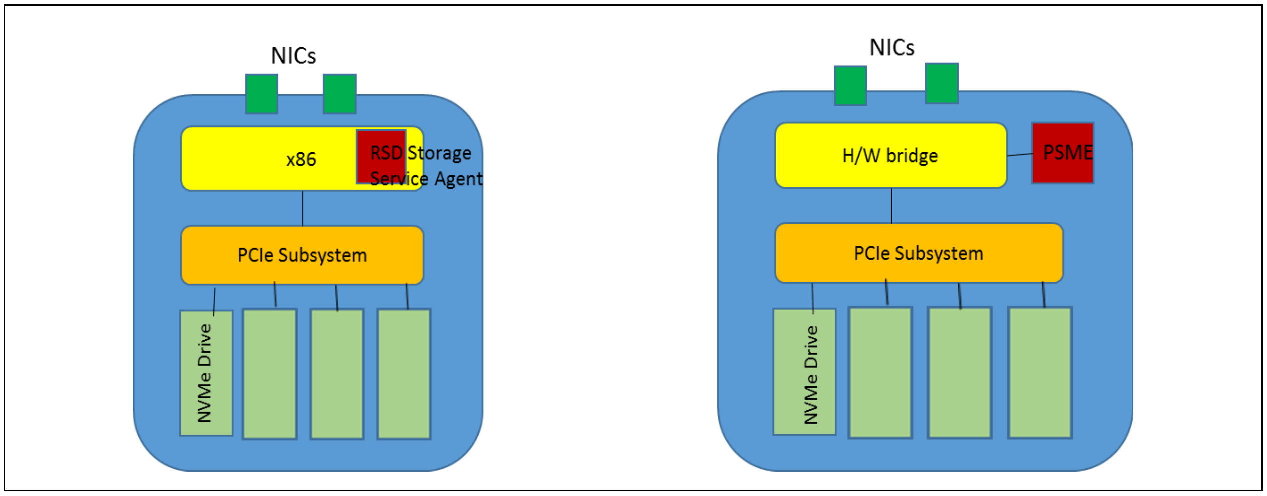

Compute
NVMe Initiator
NVMe-oF Pooled Storage Management Agent
NVMe-oF Host Interactions
Fabric
N/W configuration for Storage
Pod Manager
NVMeF Discovery Controller
NVMe-oF Pooled Storage Management Agent
NVMeF Layer
NVMe-oF Pooled Storage Management Agent
NVMe-oF Sub-system Interactions
PCIe Subsystem
NVMe Drive
NVMe-oF Discovery Sub-system Interactions

Figure 26. PSMS Realizations



12.4 High Availability NVMe-oF Use Cases

Consider High Availability (HA) in a networked storage system for all elements of the storage path, from the host to the device serving the actual data. At a rack/system level, there are three areas of redundancy that a deployment typically provides based on the requirement of the application:

- **Network** – There could be multiple paths to the data through the network to eliminate port, link, or switch failures. The NVMe specification allows for multipathing to an NVMe namespace, which is transparently extended with NVMe-oF over a fabric.
- **Chassis** – There could be redundancy in the chassis housing the data to eliminate the enclosure as a single point of failure. This is typically provided via the storage and file system services that use the NVMe devices.

For example, a Redundant Array of Inexpensive/Independent Disks (RAID) stack could be running in the host to provide replication between chassis or the host to could provide erasure coding across chassis.

- **NVMe SSD** – The most often source of failure is the storage device itself, which will require the solution to protect data from being lost in the event of a device failure. This may be delivered by the storage virtualization layer in the NVMe-oF subsystem, which manages the volume data across the physical devices using schemes such as RAID, Erasure coding and so on, or by dual ported drives.

There could be more stringent HA requirements of components within a chassis (host or storage) such as PCIe switches, NICs, and so forth.

This specification assumes both a higher layer in the host (storage, file system, and application) and storage services within the NVMe subsystem will deliver that high-availability. There will be interfaces specified to allow RSD to discover HA support in an NVMe-oF subsystem that allows for composing systems with storage HA capability.

12.5 Data Sharing with NVMe-oF

Data sharing within an NVMe-oF storage pool is defined when two or more nodes have simultaneous access to the same data on a drive, which implies access to the same namespace albeit via different controllers. NVMe does allow multiple controllers to expose the same namespace even though a single controller can only be mapped to a single host.

Supporting this configuration of namespace data sharing would require that the NVMe subsystem also support the NVMe capability of Reservations. This provides the coordination mechanism for the multiple nodes to limit which node can write data into the namespace at any given time. Refer to the NVM Express specification for details on the Reservation capability.

While the reservation capability provides a mechanism for host software to protect against racing write, Intel® RSD does not prevent multiple hosts from sharing the same data. It is up to the host software to coordinate access to the data. However, an RSD solution may choose to cancel the volume assignment to multiple hosts if the NVMe-oF target does not support reservations.

Implementation Note: The Intel® RSD v2.3 or later the PODM disallows volume sharing between multiple hosts when the attach/detach API is used. However, if the zone creation APIs are manually used, then the PSME will allow it.

12.6 Functional Requirements

The various instantiations of the NVMe Pooled Storage Management Agents will deliver the Intel® RSD experience of automated management and dynamic composition of NVMe-oF storage resources at the Rack level. Just to recap the management model described in Section 12.3.2, Logical Intel® RSD Management Model the three types of elements that require management are at the host, NVMe Target (subsystem), and network management for storage.

12.6.1 Management Functions

The primary purpose of managing NVMe-oF deployments via RSD is to provide an automatable, user-friendly way of orchestrating storage and compute to create composed nodes. The following subsections discuss the main functions that need to be considered for effective composition.

12.6.2 Services/Capabilities of Target Subsystem

Section 12.2, NVMe* over Fabric Topologies illustrates the two high-level topologies of NVMe-oF subsystems that deliver a wide range of storage services. At one end of the range, a subsystem will export physical NVMe drives (full or partial based on their PCIe controllers' capabilities) with minimal translation necessary to transport NVMe over the fabric. At the other end of the range, a subsystem could deliver advanced data services akin to high-end storage arrays. To comprehend such varying capabilities of NVMe subsystems, a notion of class-of-storage-service will be required for NVMe-oF subsystems participating in RSD deployments. The subsystem will need to provide a set of capabilities that taken together describe the class-of-service that the subsystem belongs too.

Broadly, Class of Storage Service comprises the following capabilities:

- **Role**: This specifies the role of the NVMe-oF element being managed. There are three roles that are currently defined, namely:
  - **Storage**: This indicates that this subsystem is a provider of storage data (Namespaces)
  - **Discovery**: This indicates that this subsystem is a provider of NVMe-oF Discovery Services
  - **Host:** This indicates that this element is a host (Initiator) able to connect to data and discovery subsystems
- **Storage Virtualization**: The NVMe-oF subsystem indicates whether it exports physical drives in the chassis or virtualizes the drives into volumes and exposes that. Some subsystems could do both.
- **Data Protection**: Data can be protected against physical drive failures in an NVMe subsystem via various schemes including leaving the protection to a higher layer. It follows that the subsystem shall indicate support for No Protection, Mirroring, RAID (including levels), and Erasure Coding as these schemes.
- **Data Efficiency**: Storage subsystems often employ various schemes to efficiently use the storage media. This is done both for cost (more capacity) and wears efficiency (reduce writes to the media). The subsystem shall indicate None, Deduplication, and Compression as supported schemes.
- **Endurance**: A subsystem dealing with solid-state media will have to manage its endurance. Endurance capabilities such as number of drive writes per day (DWPD) may be indicated as a metric of endurance of the subsystem.

In addition to the class of service, the subsystem should advertise the following capabilities:

- **Performance**: As with data services, there could be a wide range of NVMe-oF subsystem products that deliver different levels of performance over different vectors. Specifying a single performance number is not an easy task because it is workload and usage model dependent. However, any information that can illustrate the performance class of the subsystem can be useful for the orchestrator in choosing the right storage elements during composition. Metrics such as the Peak Input/output Per Second (IOPS) (4K, read/write), read/write latency range (idle and loaded), and Data Bandwidth are useful performance metrics that the subsystem can advertise.
- **Physical Capacity**: Total usable physical capacity and current available physical capacity.

12.6.3 Initiator (Host) and Target Volume Composition

A key aspect of system composition is to be able to assign data stores to hosts. To do that, all the NVMe subsystems must be discovered, their connectivity information must be known, and a mechanism to associate them with hosts must be provided. With that in mind, all three NVMe-oF elements have functional requirements that need to be met.

12.6.3.1 Discovery Service Management

Optional:

The Discovery Service (a special target controller defined in NVMe-oF) is the first stop for a host to connect to and find out about the target subsystems it has access to. The PODM must know about the existence of a discovery service and the details necessary to configure it. Therefore, it follows that the discovery service must specify its role, network port information, and NQN to the PODM. It also needs to take NVMe-oF subsystem and host information as input for configuring its Discovery Log Page for every host. The Discovery service can run on any machine that has access to both the in-band and OOB network. It could be hosted alongside the PODM but at this time requires RDMA support for the in-band network. A Transmission Control Protocol (TCP) binding is in the process of being proposed for NVMe-oF, which will remove the RDMA requirement once that is ratified.

12.6.3.2 Target Subsystem Management

Required:

The NVMe subsystems that provide storage (targets) contain properties that need to be discovered and configured. Such a subsystem must advertise its role, its network port information, NQN Name, physical and virtual data store information. The subsystem may export namespaces from the physical drive directly or create its own virtual namespace out of the physical drives. In either case, it may support the creation of volumes even if it is a 1:1 mapping between physical drives and volumes. The key function of the subsystem is to selectively allow hosts to connect to it and have access to a set of NVMe namespaces. This implies that it shall take the host and volume information as input to create appropriate internal records that allows the host to access the logical namespaces (volumes). The subsystem may also support configuration of its parameters such as NQN via RSD as the mechanism to bootstrap it.

12.6.3.3 Host Management

As required for Discovery Service Management and Target Subsystem Management, the host shall also specify its role, its network port information, and NQN when discovered.

12.6.3.3.1 OOB Host Configuration with Discovery Service

Optional:

As stated above, the host needs to know the identity of the Discovery Service (NQN, IP address) and a method to force it to rescan for changes in the NVMe-oF system-wide configuration (such as new volumes added/removed, targets added/removed, discovery service modified). The host shall also support configuration of its parameters such as NQN by RSD as the bootstrap mechanism for the NVMe-oF Initiator. RSD will support an OOB mechanism to configure a host with the following:

- Host’s assigned NQN
- Discovery Service’s IP address
- Discovery Service’s Transport Protocol (`“TCP”, “RoCE”, “iWARP”`)
- Discovery Service ID (Port ID).

Optionally the configuration also includes the Discovery Service’s public security key (for TLS communication between the Initiator and Discovery Controller).

The PODM will send this information to the host BMC/PSME, which will then set BIOS variables corresponding to these parameters. The NVMe-oF software Initiator will read these BIOS variables during startup and be able to participate in the NVMe-oF in-band protocol to consume storage resources on the network. The Initiator will then get all its target information (including notification of events) from the discovery controller and no more OOB configuration will be required. The Initiator may also use the Redfish Host Interface to get this information from the host BMC/PSME.

12.6.3.3.2 Unified Extensible Firmware Interface (UEFI) Variables for NVMe-oF Host Configuration

Variables are defined as key/value pairs that consist of identifying information plus attributes (the key) and arbitrary data (the value). Variables are intended for use as a means to store data that is passed between the Extensible Firmware Interface (EFI) environment implemented in the platform and EFI OS loaders and other applications that run in the EFI environment.

The following UEFI runtime services are available for managing variables.

Table 10. UEFI Runtime Services for Managing Variables

Name	Type	Description
`GetVariable`	Runtime	Returns the value of a variable
`GetNextVariableName`	Runtime	Enumerates the current variable names.
`SetVariable`	Runtime	Sets the value of the variable.
`QueryVariableInfo()`	Runtime	Returns information about the EFI variables.

Use the following Globally Unique Identifier (GUID) to associate with the NVMe-oF Variable:

```
#define NVME_OF_VARIABLE_GUID { 0xbfe2007a, 0xbcc8, 0x4184, 0x8b, 0xb7, 0xce, 0xb9, 0x2f, 0x9e,
0x4d, 0xac }
```

The Variable Data will have the following structure.

```
Typedef struct{
 UINT64                    AssignedNQN;
 UINT64                    DiscoveryServiceIPAddressLo;
 UINT64                    DiscoveryServiceIPAddressHi;
 UINT64                    DiscoveryServiceProtocol;
 UINT64                    DiscoveryServiceID;
 UINT64                    DiscoveryServicePSK;
```

`AssignedNQN` – `NQN` assigned to this host. This allows the host to be provisioned with an NQN assigned by the PODM

`DiscoverServiceIPAddressLo` – IP address of the Discovery Service. This field contains an IPv4 address or lower 64-bits of an IPv6

`DiscoverServiceIPAddresshi` – Upper 64-bits of the IPv6 address of the Discovery service, if applicable

`DiscoveryServiceProtocol` – This specifies the network protocol required to access the Discovery service. The valid values are TCP (`0x1`), RoCEv2 (`0x2`), iWARP (`0x3`)

`DiscoveryServiceID` – This specifies the network port number of NVMe-oF protocol to access the discovery service (for example, TCP port number)

`DiscoveryServicePSK` – Discovery service public key:

```
#define NVME_OF_VARIABLE_SIZE  ..........................sizeof(NVME_OF_VARIABLE_DATA)
```

The Variable will have the following Attributes:

```
#define EFI_VARIABLE_NON_VOLATILE 0x00000001
#define EFI_VARIABLE_BOOTSERVICE_ACCESS  0x00000002
#define EFI_VARIABLE_RUNTIME_ACCESS  0x00000004
#define NVME_OF_VARIABLE_ATTRIBUTE                    (EFI_VARIABLE_NON_VOLATILE |
EFI_VARIABLE_BOOTSERVICE_ACCESS | EFI_VARIABLE_RUNTIME_ACCESS)
```

The Variable will have the following Name:

```
#define NVME_OF_VARIABLE_NAME   "NVMeOF Config Params"
```

Use the `GetVariable()` and `SetVariable()` UEFI runtime services using the following parameters

```
GetVariable()
Prototype
Typedef
EFI_STATUS
GetVariable(
         IN CHAR16          *VariableName,
         IN EFI_GUID                   *VendorGuid,
         OUT UINT32         *Attributes,
         IN OUT UINTN       *DataSize,
         OUT       VOID                *Data
);

SetVariable()
Prototype
Typedef
EFI_STATUS
SetVariable(
         IN CHAR16          *VariableName,
         IN EFI_GUID                   *VendorGuid,
         IN UINT32          *Attributes,
         IN UINTN           *DataSize,
         IN VOID                       *Data
);

Parameters
VariableName       NVME_OF_VARIABLE_NAME,
VendorGuid                 NVME_OF_VARIABLE_GUID,
Attributes                 NVME_OF_VARIABLE_ATTRIBUTE,
DataSize                   NVME_OF_VARIABLE_SIZE,
Data                       NVME_OF_VARIABLE_DATA
```

12.6.3.3.3 I/B Host Configuration

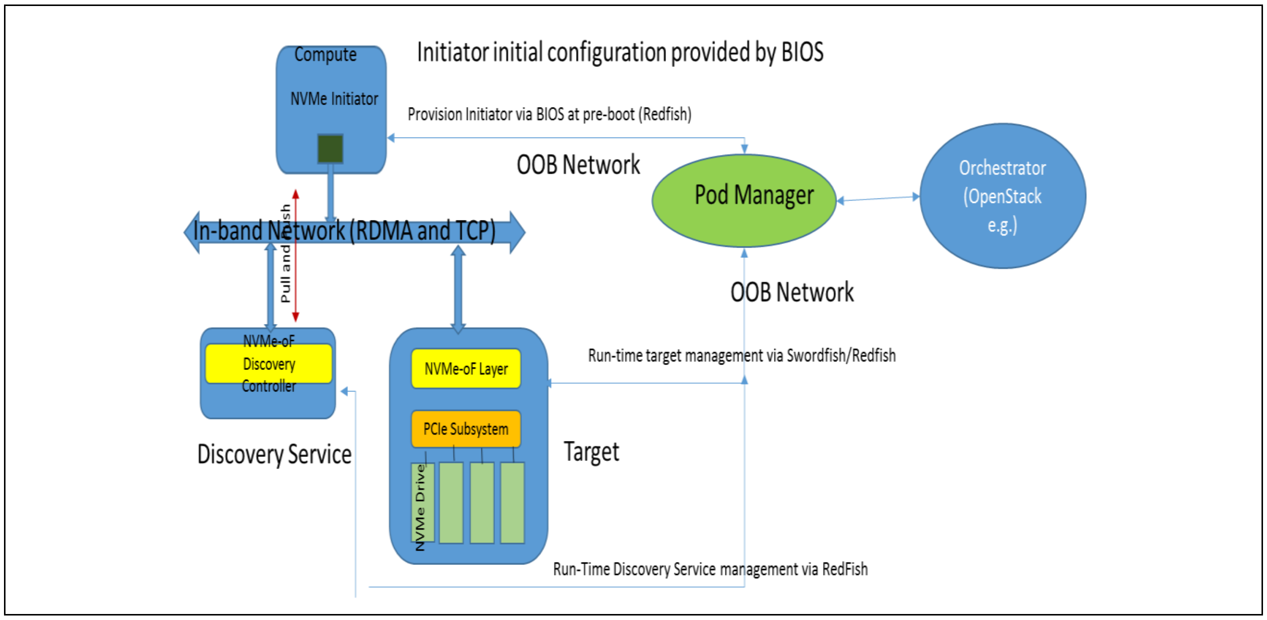
Note: Required if OOB via BIOS method is not supported.

If a discovery controller is not being used, an in-band agent is required on the host to communicate with the PODM using the initiator Redfish interface. This will require the PODM to explicitly specify the target information for every target that is assigned to the host.

The host agent can use SSDP to locate the PODM and TLS to authenticate it.

If a discovery, service is being used and the OOB mechanism is not present (described in Section 12.6.3.3.1, OOB Host Configuration with Discovery Service), an in-band mechanism can be used to discover the discovery service on the network. This can be done by adding a DNS entry for the well-known Discovery NQN in a local DNS server. The initiator can connect to it using a default protocol and port number as specified in Table 3, *NVM Express over Fabrics Specification*. For example, it can use `RoCev2` as the default protocol and 4791 as the default port. If that fails, then it can have secondary default protocol of TCP and the port number as specified in the NVMe-oF TCP transport binding specification. The initiator can also support a mechanism to configure the network information in a file to override the default.

Figure 27. Initiator, Target, and Discovery Management Roles


Compute
NVMe Initiator
Initiator initial configuration provided by BIOS
Provision Initiator via BIOS at pre-boot (Redfish)
OOB Network
Pod Manager
Orchestrator (OpenStack e.g.)
In-band Network (RDMA and TCP)
Pull and Push
OOB Network
NVMe-oF Discovery Controller
NVMe-oF Layer
PCIe Subsystem
NVMe Drive
Run-time target management via Swordfish/Redfish
Discovery Service
Target
Run-Time Discovery Service management via RedFish

12.6.4 Security

Management of storage within RSD has several security implications that must be addressed:

- Protecting the data on the drive from unauthorized access
- Protecting the NVMe-oF Target subsystem from unauthorized access from a compute host
- Protecting the Target subsystem from unauthorized access via the Intel® RSD management function.

12.6.4.1 Data Security

The data on the physical drive must be secure from unauthorized accesses. There are two scenarios that need to be addressed for this case, namely, data at-rest protection and data erase.

12.6.4.1.1 Data Erase Support

Required:

One of the security issues in a dynamically composed multitenant system is that the same data area in a disk may be successively owned by different applications/users. This requires the data in the volume being unassigned from a host be erased before putting the volume back into the pool for subsequent assignment. However, in case of a reassignment to the same application on a different host (due to workload migration), the data should not be erased.

The PODM must inform the PSMS of the need to erase volume data during an unassigned request. The default policy in the PODM shall be to erase the drive during decomposition unless told to reserve it for reassignment. If the volume is entirely mapped to a single drive, the PSMS must secure erase the drive. The PSMS must erase all security keys associated with the drive as well during this process. If the volume is only partially composed from a drive, then it must be zeroed out efficiently (using TRIM for example). The PSMS can do this via the Admin Queue of the drive directly or indirectly via the NVMe-oF target.

If a volume is deleted, then PSMS shall secure erase it (if it is the entire drive) or zero it out (if partial drive).

12.6.4.2 NVMe-oF Target PSME (PSMS) Access Security

Required:

NVMe-oF enables a storage device to be visible and accessible on the network by a multitude of hosts and applications. This requires the target have an interface that allows it to be provisioned with volume, access control lists for hosts and so forth. This interface must be secured by authentication, confidentiality, and integrity mechanisms. This will follow the Intel® RSD security model where the PSMS must conform to any role-based security requirements of the PODM as well as be able to authenticate the PODM.

Access to NVMe-oF target management APIs is only allowed with TLS. This means that the PODM and NVMe-oF target must establish a TLS connection before they can communicate. Establishing this TLS connection requires mutual authentication between the PODM and the selected NVMe-oF target. Once authentication succeeds, this TLS connection between the PODM and NVMe-of target will provide integrity and confidentiality.

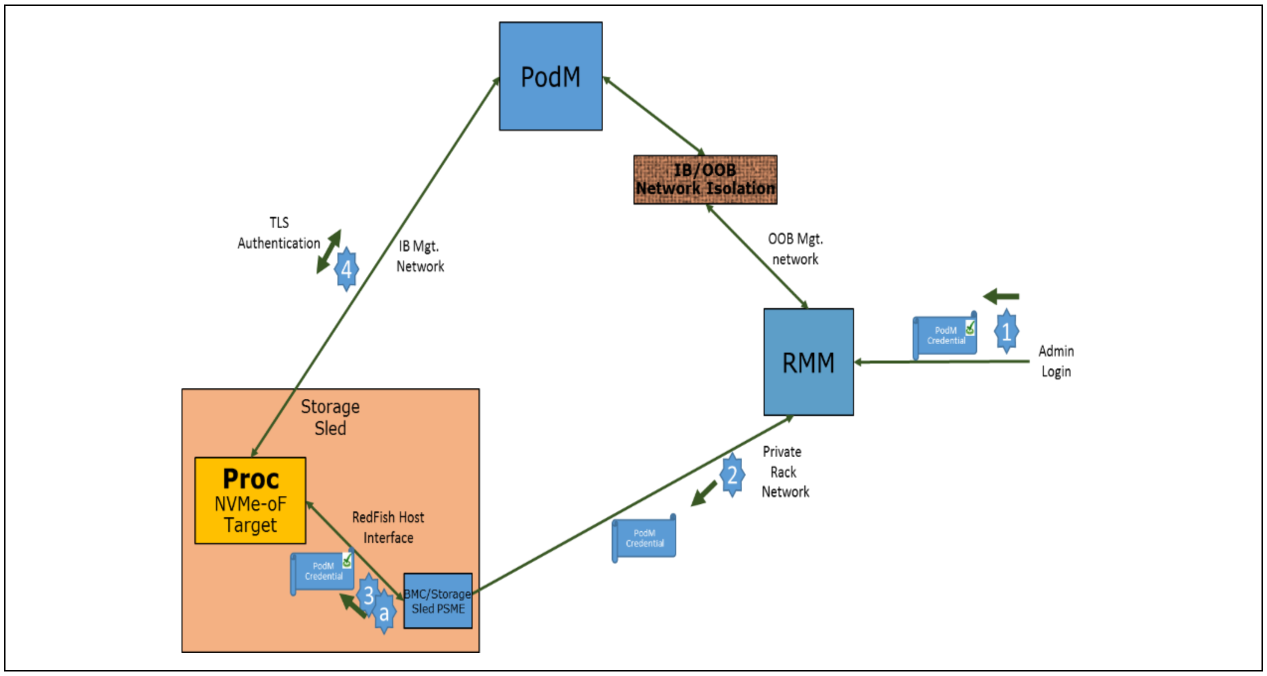
Intel® RSD will support this mutual authentication between the PODM and NVMe-oF target/discovery service by enabling provisioning of credentials to both the PODM and NVMe-oF target as described in the following subsections.

12.6.4.2.1 PODM Credential Provisioning to NVMe-oF Target

1. The PODM authentication credential is provisioned to a rack (RMM) during initial rack set up.
2. During rack initialization, or when a new sled joins an already running rack, the RMM passes this credential to each sled's PSME using the rack private network
3. The NVMe-oF Target BMC can pass this credential to NVMe-oF target software in one of the following ways:
    a. Wait for NVMe-oF target SW to use Redfish Host Interface functionality to retrieve PODM credential.
    b. Place this credential in a BIOS location where NVMe-oF target SW can retrieve it (using a BIOS call) once it is up and running.
4. The NVMe-oF target will use this credential to authenticate PODM as part of the TLS handshake.

Figure 28 illustrates the provisioning steps listed above.

Figure 28. Provisioning of PODM Credential to NVMe-oF Target



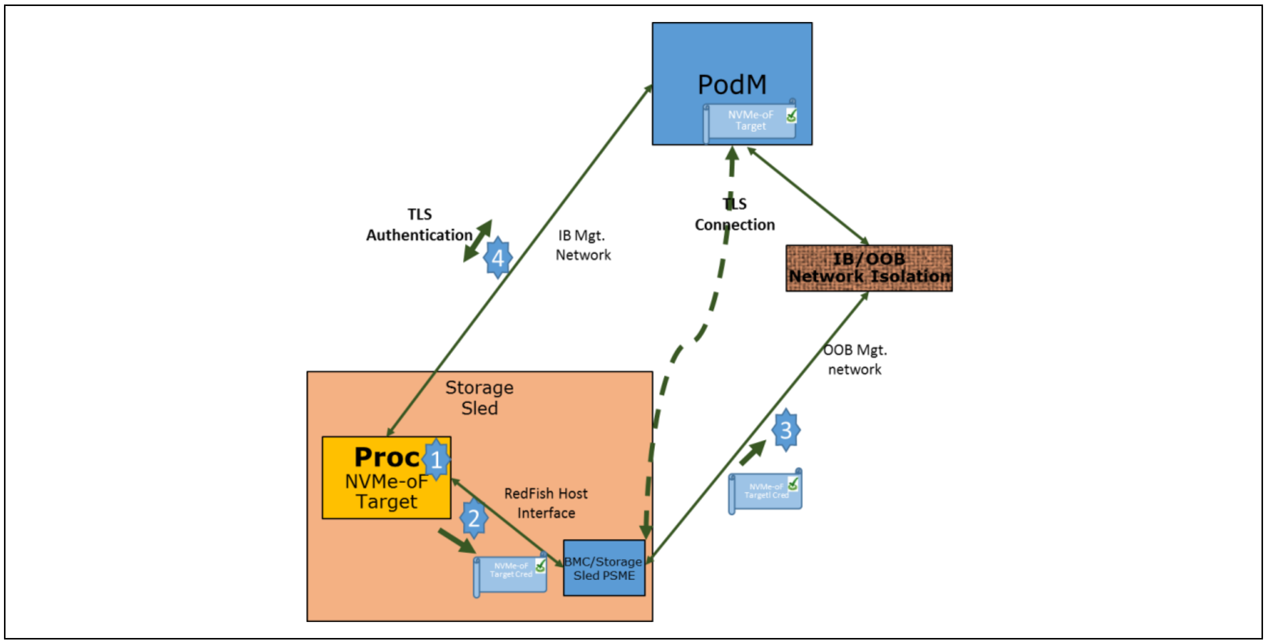
12.6.4.2.2 NVMe-oF Target Credential Provisioning to PODM

Steps required to pass a public key credential to the PODM are listed below:

1. Target NVMe-oF SW generates private/public keypair that will be used for NVMe-oF security.
2. Public key credential is passed to the NVMe-oF BMC using Redfish Host Interface functionality.
3. The PSME passes the public key credential to the PODM OOB using the OOB management network and secure connections created by the RSD management infrastructure (i.e., PSME to PODM TLS connection).
4. The PODM and NVMe-oF target PSME can now communicate securely using TLS.

The PODM will use this credential to authenticate the NVMe-oF target as part of the TLS mutual authentication. Figure 29 Illustrates the steps listed above.

Figure 29. Provisioning of NVMe-oF Target Credential to PODM



12.6.4.3 NVMe-oF Volume Access Security

In addition to controlling the host access permission to volumes, the target may require a host to have secure access to a volume, which implies strong authentication, integrity, and encryption mechanisms between the host and target. The *NVM Express over Fabrics specification* (refer to Table 3) supports two methods of authentication, which can be used together – Fabric Secure Channel and NVMe In-band Authentication.

12.6.4.3.1 Fabric Secure Channel

Optional:

The Fabric Secure Channel is the method of establishing a secure connection using transport-specific security schemes such as IPSec (ESP in Transport Mode). If such schemes are used and required by a subsystem, then the credentials may be provisioned by RSD if the specific solution demands it.

In the case of Fabric Secure Channel, NVMe-oF targets will support Encapsulating Security Payload (ESP), which provides authentication, integrity, and confidentiality protection, in Transport Mode (i.e., IPsec for point-to-point communication between initiator and target).

RSD will support IPsec (ESP in Transport Mode) by enabling the use of the Intel® RSD management infrastructure to perform secure credential provisioning for both target and initiator. An NVMe-oF target must be securely provisioned with an NVMe-oF initiator credential (for example, a public key of the initiator that target will serve), and an NVMe-oF initiator must be securely provisioned with an NVMe-oF target credential (for example, a public key for the target serving that initiator), so that they can authenticate each other and create an IPsec session. Intel® RSD will support the previously mentioned credential provisioning as described in the following subsections.

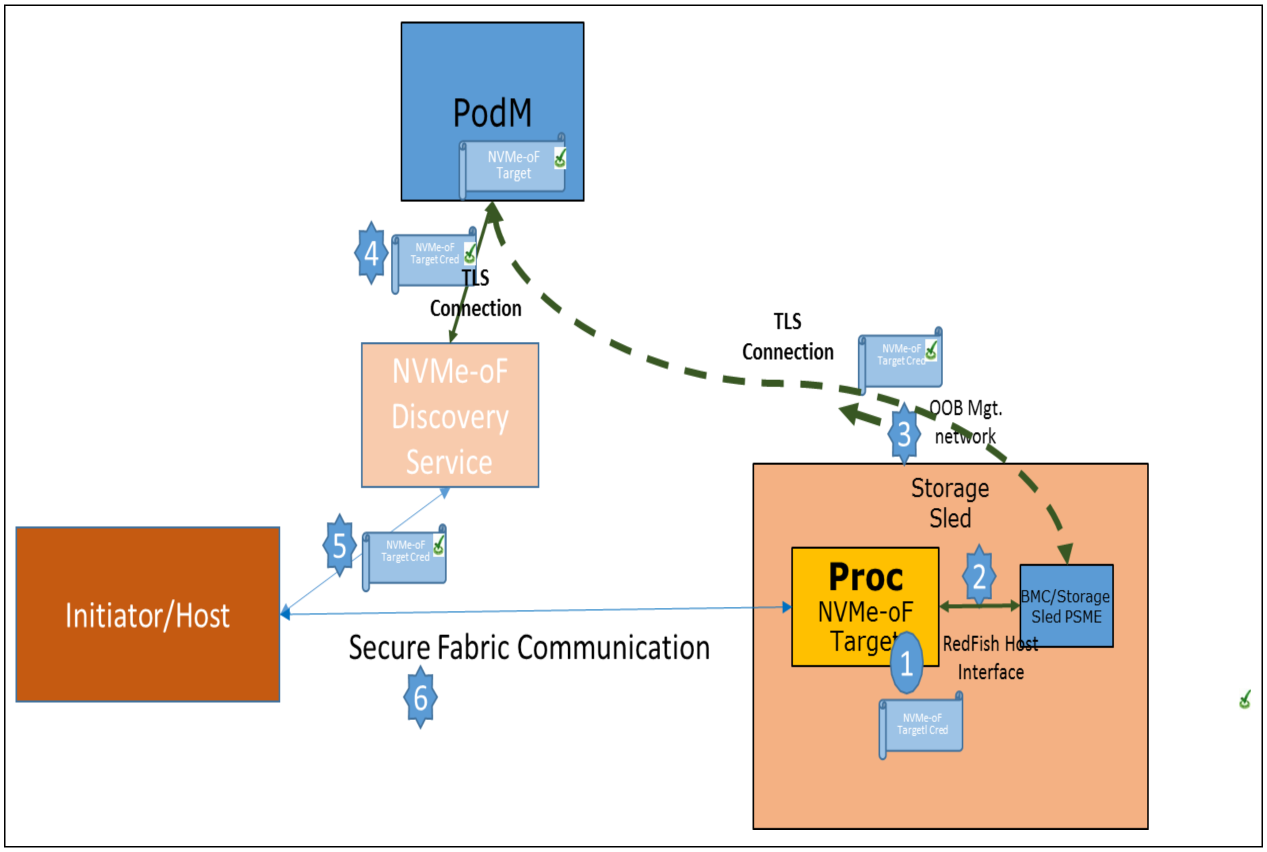
NVMe-oF Target Credential Provisioning to NVMe-oF Initiator

The following steps describe how this provisioning is accomplished:

1. NVMe-oF target starts (for the first time) and generates a public/private keypair to be used for NVMe-oF security.
2. NVMe-oF target uses Redfish Host Interface to pass its public key credential to its BMC/PSME.
3. BMC/PSME passes this credential to PODM using OOB management network and secure connections created by RSD management infrastructure (i.e., PSME to PODM TLS connection).
4. Later (for example, at node composition time) PODM provisions this credential to NVMe-oF discovery service (just an NVMe-oF target without storage) using its secure management connection to the Discovery Target.
5. NVMe-oF initiator can then pick up the NVMe-oF target credential as part of the NVMe-oF discovery process.
6. Secure Fabric communication enabled between initiator and target.

Figure 30 depicts the above flow. The discovery service log entry will need to contain the target public key, which is 256B. This will be an enhancement of the discovery log and is not defined in v1.0 of the *NVM Express over Fabrics Specification*, refer to Table 3.

Figure 30. Target Credential Hand-off to Initiator for Secure Fabric Comm (IPSec)



NVMe-oF Initiator Credential Provisioning to NVMe-oF Target

The following steps describe how this provisioning is done:

1. NVMe-oF initiator starts (for the first time) and generates a public/private keypair to be used for NVMe-oF security.
2. NVMe-oF initiator uses Redfish Host Interface to pass its public key credential to its BMC.
3. BMC passes this credential to PODM using OOB management network and secure connections created by RSD management infrastructure (i.e., PSME to PODM TLS connection).
4. PODM provisions this credential to NVMe-oF target that will service this initiator using its secure management connection to the Target.

12.6.5 Telemetry

The physical components (CPU, network ports, and so on) in a target system should support the Redfish OOB telemetry interface defined for those components. In addition, the target should export telemetry on a per volume basis.

12.6.5.1 Volume Telemetry

Required:

The Target should report the total.

Optional:

The Target should report the total and used capacity of every volume.

12.6.5.2 NVMe SSD Telemetry

Required:

NVMe SSDs support NVMe SMART metrics that should be made available through the RSD from the NVMe-oF systems. Currently, NVMe SMART provides the following metrics. Refer to Table 3, *NVM Express over Fabrics Specification*, for details and updates.

- Temperature of the device is measured in Kelvin
- % Available Spare – This gives an indication of the SSDs ability to recover from NAND cell errors
- %User – This gives an estimate of the lifetime used of the SSD. This value can go up to 255%
- Data Units Read/Written – This is reported in 512B units since the beginning of the drive's use. This can be used to compute the average bandwidth by polling the SSD at regular intervals.
- Host Read/Write Commands – This is reported by the number of commands issued to the SSD controller by the host. This can be used to compute the average host command rate by polling the SSD at regular intervals.
- Controller Busy Time – This reports the number of minutes the controller has been busy processing commands. This combined with the power-on hours can give a sense of the percent of time the controller is being used.
- Power Cycles
- Power-on Hours
- Unsafe Shutdowns
- Media Errors.

12.6.6 Storage Device Management

The NVMe devices in the target can be managed by both in-band (PSMS) and OOB (PSME) mechanisms. The NVMe origination has defined a management interface, NVMe-MI, to manage the device via OOB mechanisms (like BMC or PSME). If the target enclosure were to have OOB management of the devices, then it needs to be coordinated with the PSMS for reconciliation of discovered resources as well as for performing disruptive operations like firmware updates, secure erase, and so forth.

The following NVMe-MI functions may be relevant to RSD if used directly on the device via OOB (SMBus):

- Management Commands: Controller Health status poll - Percentage Used (`PDLU`), Critical Warning (`CWARN`)
- Admin Commands via NVMe-MI:
  - Non Interfering with host - Identify, Get Log Page
  - Interfering with the host – Firmware Image Download, Firmware Activate/Commit
  - Controller Metadata – Get OS Driver name and version
  - Namespace Metadata – OS Name.

The PSMS should provide device management feature if the chassis does not provide them. These should include:

- Device Health management and monitoring
- Device Firmware management
- Telemetry Information.

The PSMS must be allowed to issue admin commands to the NVMe drives to update the firmware and extract telemetry such as NVMe SMART. In addition, the NVMe-oF target must prevent firmware update and namespace creation commands from the host from reaching the drives.

12.6.7 Hot-Plug

There are two cases where hot-plug is relevant in an NVMe-oF deployment.

Hot-Plug at the Host

NVMe-oF volumes in the host may be dynamically added and removed while the OS is running (equivalent to hot add/remove). Since these volumes are usually virtual volumes, a modern OS stack provides the means to notify the stack of arrival/departure of volumes.

For software initiators, this scheme will already be in use because it relies on these OS interfaces to surface all volumes. If the NVMe-oF volumes are surfaced as drives by a host bus adapter (HBA) (hardware initiator), then the HBA driver must use the OS defined mechanisms to create/remove NVMe-oF volumes. The actual hot-plug of a physical HBA will require PCIe hot-plug support from the host platform.

Hot-Plug at the Target

If a target platform supports physical NVMe drive hot-plug, then the PODM must be notified by the target PSMS of the arrival/departure of the drives. The PSMS can use the existing RSD state change notification API to notify the PODM of a state change. The PODM must then query the PSMS and determine the addition/loss of drives in the system.

Note: The requirements and mechanism used by a target to support NVMe drive hot-plug is implementation-specific and is beyond the scope of this specification.

12.7 Intel® RSD Management Software for NVMe-oF

12.7.1 Software Architecture

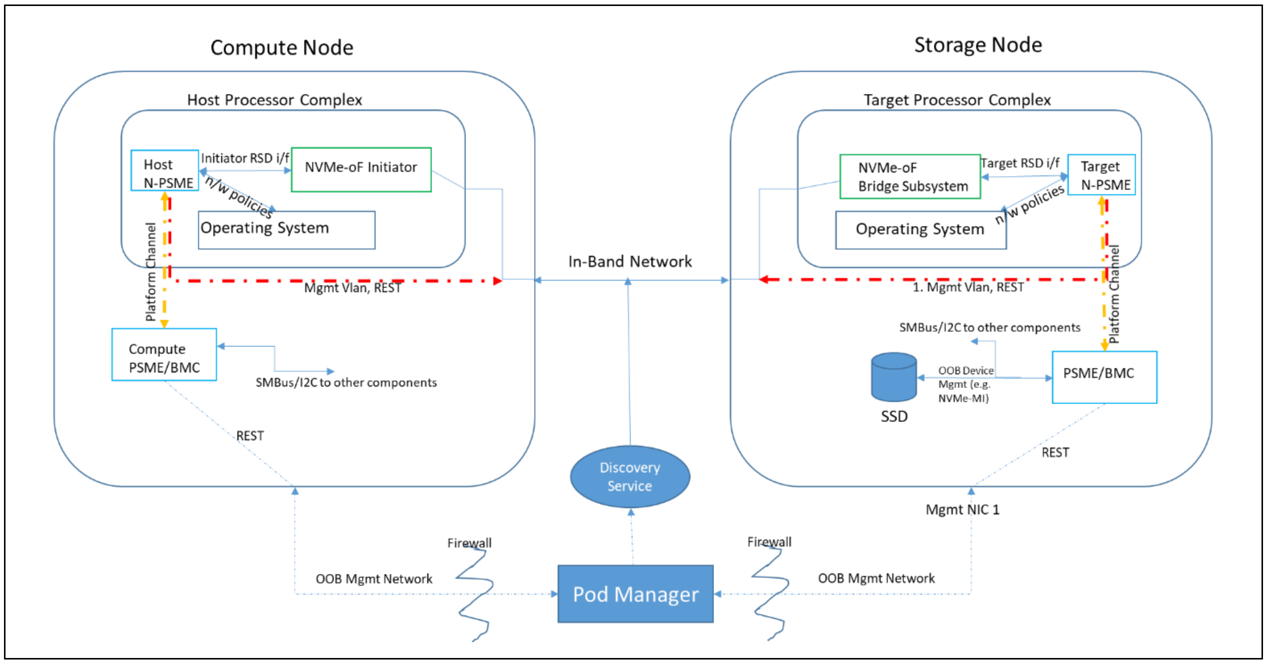
The software components needed for creating an Intel® RSD compliant NVMe-oF deployment are shown in Figure 31. The general implementation of the NVMe target or initiator will be in a processor complex, running within an OS. While part of the target or initiator may be implemented in hardware for performance reasons, there will be a processor complex running the NVMe-oF management functionality. That processor complex is depicted in Figure 31 for both the host and the target running an operating system and the NVMe-oF software.

The NVMe-oF Pooled Storage Management Service (Target PSMS) will run on the same control processor complex (as shown in Figure 31) and interface with the NVMe-oF target subsystem to get and set the relevant information needed to manage it. This Target PSMS will be an Intel® RSD component. It is expected that this processor complex will not have direct access to the private OOB management network. This will require the PSMS to communicate with the PSME/BMC to be provisioned with credentials for establishing a connection to the PODM over the in-band network using the APIs in the Intel® RSD Specifications. Refer to the Note in Table 3.

The Initiator can be software-based (running on the host CPU) or hardware-based (HBA). The initiator needs to be configured with its NVMe-oF credentials to allow it to establish connections with NVMe-oF targets. For an HBA model, the compute PSME residing on the platform on the OOB network can configure the initiator using a management interface like Management Component Transport Protocol (MCTP).

For SW Initiators (typically the NVMe driver in the OS), an agent (host PSMS) will run on the host to configure the initiator with its storage target information. This host PSMS will require information from the PODM, which can be done directly or indirectly. The indirect communication can be done using the compute PSME/BMC to communicate with the agent running via BIOS runtime variables (refer to Section 12.6.3.3.1, OOB Host Configuration with Discovery Service). Direct communication requires the host PSMS to communicate with the PODM through the PSME by using the Redfish Host Interface (refer to Section 12.6.3.3.3, I/B Host Configuration).

Figure 31. Intel® RSD NVMe-oF Pooling SW Components



12.7.2 NVMe-oF Target Management Functional Interface

Required:

This section lists the Intel® RSD management interface requirements between the NVMe-oF Subsystem and the PSMS that will be required to deliver RSD management. Some of the interface elements are optional because they are dependent on the level of functionality of the NVMe-oF target.

The actual realization of the interface between the PSMS and the NVMe-oF subsystem is implementation-dependent and can be a combination of configuration files, CLI and/or IPC.

Table 11. NVMe-oF Target Interface Requirements

Function	Input	Output	Required/ Optional	Comments
Get Service Role	Subsystem NQN	Storage	Required	Provider of Data Service
Get Subsystems	None	List of Subsystem NQNs	Required	If already configured
Get Network Ports	None	List of Port IDs	Required	PSMS generates unique ids for all the in-band data plane Ethernet ports
Get Network Port Info	Port ID	Transport Address, Type, Service Identifier	Required	Address: String Type: "`RoCE`", "`iWARP`", "`FC`", "`TCP`"
Get Supported NVMe-oF Controller Type	None	Controller Type	Required	This would return "`Static`" or "`Dynamic`". Most targets are expected to implement Dynamic
Get Capability	Capability Type: Virtualization	Full Drive, Partial Drive, Virtual Namespace	Required	**Full Drive** implies the subsystem can export a full drive. **Partial Drive implies** it can export multiple namespaces from a single drive. **Virtual Namespace** implies it can export a namespace that is abstracted from the actual physical drives that is, arbitrary volumes like fdsk partitions for example.
Get Capability	Capability Type: Data Protection	RAID (Mirror, Striped, Striped with protection), Erasure Coding, Other, None	Required	Indicates data protection against disk failures managed by this storage system (within the storage box)
Get Capability	Capability Type: Data Efficiency	Deduplication, Compression, None	Required	Indicates data storage efficiency schemes employed by this storage service.
Get Capability	Capability Type: Authentication Level: Transport, NVMe	Fabric Secure Channel (IPSec, TLS, Other), NVMe in-band, None	Required	-
Get Capability	Capability Type: Performance	Peak IOPs, Rated Read and Write Latency	Required	-
Get Capability	Capability Type: Provisioning	Thin, Full	Optional	Full is the default

Function	Input	Output	Required/ Optional	Comments
Get Subsystem capability	None	Max number of physical drives, max number of vol., number of n/w ports, other NVMe capabilities	Required	This exposes the maximum capabilities of a subsystem in a target with respect to the number of devices, volumes, ports it can support. A return of 0 for any of these values implies no limit. This is based on the target system's design limitations due to say amount of RAM in the storage sled. RSD can use this to ensure correct provisioning of the subsystem.
Get Subsystem Properties	NQN	Namespace ids, port IDs, Admin Max Submission Queues Size, Security Credentials	Required	Security Credential is the public key used by the subsystem for transport channel secure communication.
Get Physical Namespaces/Drives	None	Physical Namespace (drive) GUIDs, drive serial number	Required	This is essentially physical drives or partitions as seen by the PSMS/Target.
Get Physical drive Info	Drive ID	Rated Capacity, Media Type, Namespaces, Serial number	Required	Media Type – NAND, 3DX. Some of the information is obtained from the NVMe inquiry command.
Get Physical Drive Statistics	Drive ID	Used Capacity, NVMe SMART	Required	These are physical drive statistics.
Get Assignable Namespaces	None	List of Namespace GUIDs that are assign-able	Required	These are volumes that are assignable, This namespace ID should be the same that is returned from Identify Namespace NVMe command. This will allow easy association of volume list in PODM with the volumes seen by the application running on the host node.
Create Assignable Namespace	Required Capacity, list of physical namespaces (drives), protection policy	Namespace GUID	Optional	This is required only if the subsystem supports virtual Namespaces. This is equivalent to creating a volume. PSME should generate the volume ID (namespace ID) as a GUID that is globally unique. This namespace ID should be the same that is returned from Identify Namespace NVMe command.
Get Namespace properties	Namespace GUID	Capacity, List of Physical namespaces (drives)	Required	-
Get Namespace Statistics	Namespace GUID	IOPs, read/write distribution, average I/O size, Used capacity	Optional	These are volume statistics
Get Subsystem Statistics	Subsystem NQN	Connected Host list	Required	-

Function	Input	Output	Required/ Optional	Comments
Allocate Physical Drives	NQN, List of drives	None	Optional	This is required for subsystems that do not auto-discover physical drives and start managing them.
Configure NQN	NQN	None	Required	RSD shall generate the NQN using the UUID of the Target PSME in the following format: `nqn.2014-08.org.nvmexpress:NVMf:uuid:<assigned uuid>`
Configure Network Ports	NQN, list of n/w ports, Transport addresses	None	Optional	This is required for subsystems that do not auto discover and auto-configure network ports.
Configure Network Port ID	n/w port, NVMe-oF Port ID	None	Required	A 16-bit Port ID shall be generated unique to an NQN (a simple counter will work)
Assign Namespace to Host	Subsystem NQN, Host NQN, Namespace ID, Access Rights (read/write)	None	Required	Namespace ID is one of the list of ids returned from the Get/Create Assignable Namespaces call(s)
Erase Namespace Data	Namespace ID	None	Required	Securely erases or zeroes out the namespace
Get Error Logs	Subsystem NQN	Log Page	Required	–
State Change Notification	None	None	Required	This allows the target to notify the PODM about changes in its state such as drive hot-plug

12.7.3 NVMe-oF Discovery Service Management Functional Interface

The Discovery Service is a special purpose NVMe subsystem that solely provides discovery controllers. This can optionally be implemented as part of the PODM. It follows that its ***RSD management*** interface requirements are a subset of the target ***management*** interface (refer to Section 12.7.2, NVMe-oF Target Management Functional Interface) with a few additions. Table 12 lists the required interfaces for a Discovery Subsystem.

Table 12. NVMe-oF Discovery Service Interface Requirement

Function	Input	Output	Required/Optional	Comments
Get Service Role	Subsystem NQN	Discovery	Optional	Provider of Discovery Service
Get Subsystems	None	List of Subsystem NQNs	Optional	Should be a list of 1. If primary service then it should return the NQN `nqn.2014-08.org.nvmexpress.discovery`
Get Ports	None	List of Port IDs	Optional	–
Get Port Info	Port ID	Transport Address, Type, Service Identifier	Optional	Address: String Type: “`RoCE”, “iWARP”, “FC”, “TCP”`
Get Subsystem Properties	NQN	Namespace ids, port IDs, Admin Max Submission Queues Size, Security Credentials	Optional	Namespace ID list will be 0. Security Credential is the public key used by the subsystem for transport channel secure communication

Function	Input	Output	Required/Optional	Comments
Configure Network Ports	List of n/w ports, Transport addresses	None	Optional	This is required for Discovery subsystems that do not auto discover and auto-configure network ports.
Configure Network Port ID	n/w port, NVMe-oF Port ID	None	Optional	A 16-bit Port ID shall be generated unique to an NQN (a simple counter will work)
Get Discovery Log	Host NQN	List of Discovery Entries	Required	A Discovery Entry shall contain a Subsystem NQN, Subsystem Transport Address (IPv4 and v6). Max Admin Queue Size of the subsystem, Transport Service ID, Port ID of the Subsystem, controller ID (`0xFFFF` or `0xFFFE` for dynamic controllers), Subsystem Public Key
Set Discovery Entry	Host NQN, Discovery Entry	None	Required	This is used by the PODM to set the discovery log entry for a host that needs to be associated with a target due to a composition request. Same Discovery Entry content as above.

12.7.4 NVMe-oF Initiator Management Functional Interface

Required:

The primary requirement on the host side is to configure the initiator with the location of the Discovery Service at which point it is enabled to discover all the subsystems available to it. If a Discovery Service is not available, then RSD needs to explicitly configure the host for every target. The following table lists the Intel® RSD management interfaces an Initiator needs to support.

Table 13. NVMe-oF Host Interface Requirements

Function	Input	Output	Required/Optional	Comments
Get Service Role	NQN	Host	Required	Initiator
Get Host NQN	None	List of NQNs	Required	Should be a list of 1 in the general case
Configure Host NQN	NQN	None	Required	RSD shall generate the NQN using the UUID of the Initiator PSME in the following format: `nqn.2014-08.org.nvmexpress:NVMf:uuid:<assigned uuid>`
Set Discovery Info	Discovery Service Info	None	Required	The input will contain the Discovery NQN, Transport Address of the Discovery Controller, Transport Protocol Type (RDMA, TCP and so on) and Transport Service ID (Port number) of the Discovery Controller
Discovery Log Change Notification	Discovery Service Info	None	Optional (if Discovery Service does not support in-band notification)	Indicates to the host that a discovery log state change has occurred

Function	Input	Output	Required/Optional	Comments
Connect To Target	Target NQN, Host NQN, Target Transport Info (protocol type, port), and flags (persistent and so on)	Success/ Failure	Optional (if Discovery Service is not present/used)	Indicates to the host that it should connect to a target and enumerate devices. Persistent flag indicates whether the connection should be persistent across host power cycles.

12.7.5 Network QoS Support for NVMe* over Fabric

Optional:

There are three elements to manage when it comes to delivering network QoS for NVMe-oF storage I/O. All three entities require appropriate interfaces to allow configuration of QoS policies.

12.7.5.1 Host and Target

Set Network Port DCB Settings by doing the following:

- Enable DCB on the NIC
- Configure PFC for NVMe-oF TC
- Configure priority to traffic map for NVMe-oF traffic (using port number)
- Configure BW percentage for NVMe-oF TC if desired.

12.7.5.2 Switch

Set Switch DCB settings by doing the following:

- Configure PFC for the NVMe-oF traffic class
- Configure bandwidth percentage on priority group that includes the NVMe-oF traffic class
- Enable DCBx.

12.7.6 Error Handling for NVMe* over Fabric

There are two categories of errors that RSD will need to manage:

1. (**Required**) RSD Operations Errors – These are errors encountered during operations, such as Composition/Decomposition. These should be handled using existing RSD mechanisms.
2. (**Optional**) NVMe-oF errors - NVMe-oF specifies a set of error conditions whose status would be useful to capture and report. These errors can be reported via the LogService collection resource of a storage system. Some specific errors in this category could be:
    a. Connection errors – Report reasons such as transport failures
    b. Error status of fabric completion commands (as specified in Section 2.2.1 of the *NVM Express over Fabrics Specification* refer to Table 3.)

12.7.7 Common Configuration Flows for NVMe* over Fabric

Figure 32 and Figure 33 show example functional flows required to configure initiators and targets as well as a discovery controller (if it exists). The non-discovery controller flow in Figure 33 may be the implementation of RSD if a Discovery Controller (Service) is not included in the PODM due to lack of support for Thermal Design Power (TCP) in the Discovery Controller.

Figure 32. Target and Initiator Configuration using NVMe-oF Discovery Service

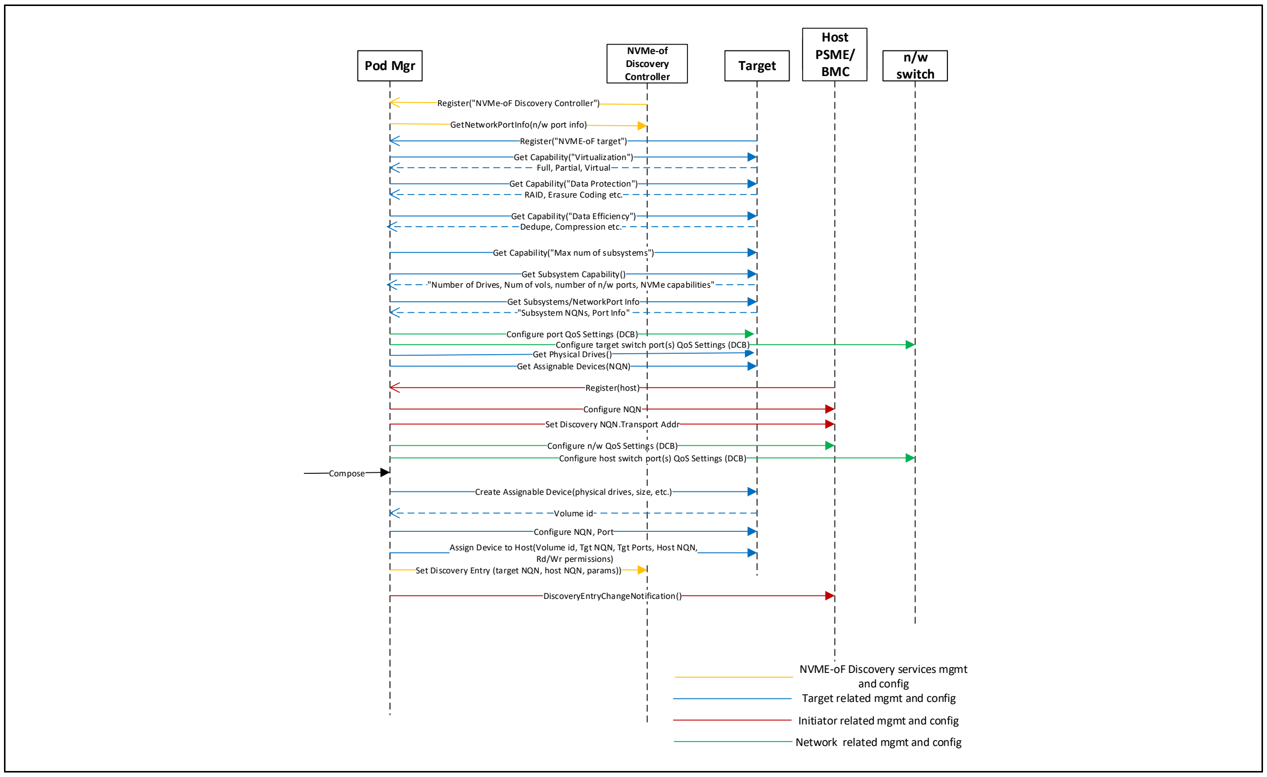
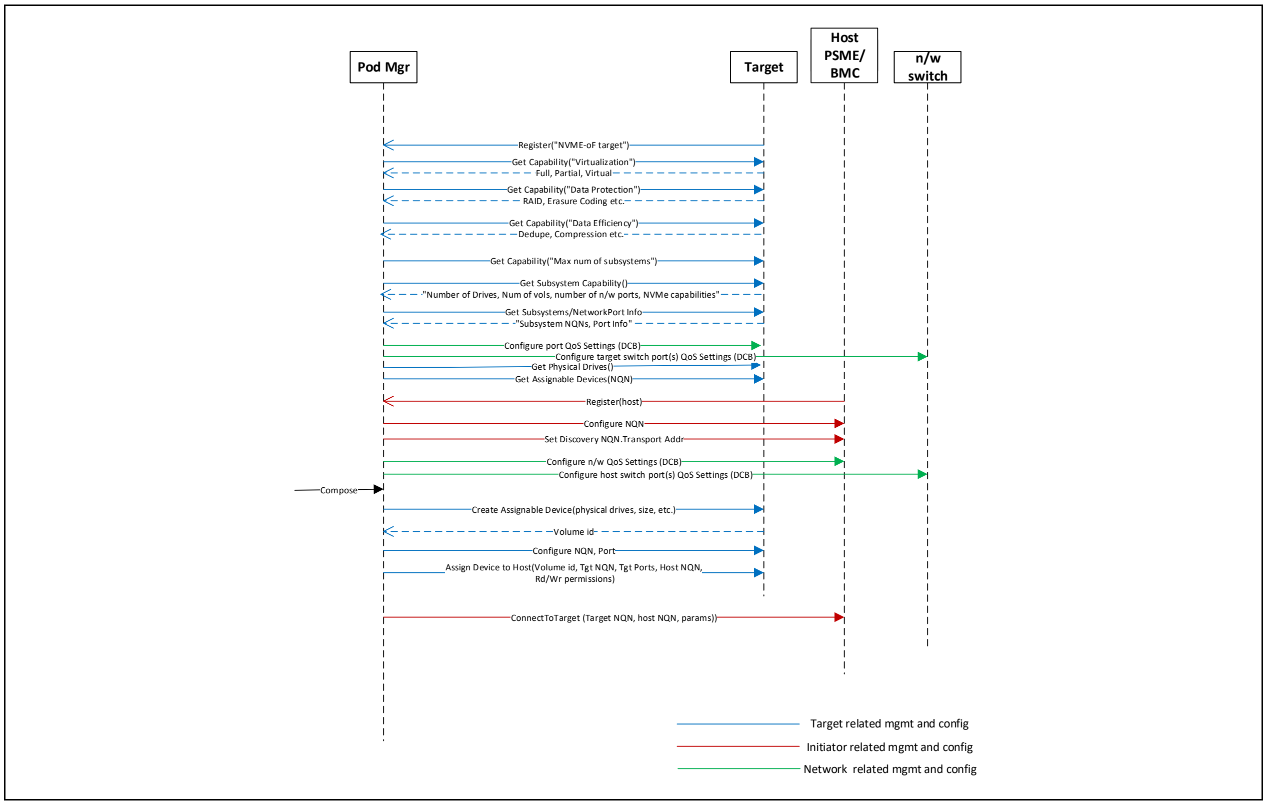

Pod Mgr
NVMe-of Discovery Controller
Target
Host PSME/ BMC
n/w switch
Register("NVMe-oF Discovery Controller")
GetNetworkPortInfo(n/w port info)
Register("NVMe-oF target")
Get Capability("Virtualization")
Full, Partial, Virtual
Get Capability("Data Protection")
RAID, Erasure Coding etc.
Get Capability("Data Efficiency")
Dedupe, Compression etc.
Get Capability("Max num of subsystems")
Get Subsystem Capability()
"Number of Drives, Num of vols, number of n/w ports, NVMe capabilities"
Get Subsystems/NetworkPort Info
"Subsystem NQNs, Port Info"
Configure port QoS Settings (DCB)
Configure target switch port(s) QoS Settings (DCB)
Get Physical Drives()
Get Assignable Devices(NQN)
Register(host)
Configure NQN
Set Discovery NQN.Transport Addr
Configure n/w QoS Settings (DCB)
Configure host switch port(s) QoS Settings (DCB)
Compose
Create Assignable Device(physical drives, size, etc.)
Volume id
Configure NQN, Port
Assign Device to Host(Volume id, Tgt NQN, Tgt Ports, Host NQN, Rd/Wr permissions)
Set Discovery Entry (target NQN, host NQN, params))
DiscoveryEntryChangeNotification()
NVME-oF Discovery services mgmt and config
Target related mgmt and config
Initiator related mgmt and config
Network related mgmt and config

Figure 33. Target and Initiator Configuration without Using NVMe-oF Discovery Service


Pod Mgr
Target
Host PSME/ BMC
n/w switch
Register("NVME-oF target")
Get Capability("Virtualization")
Full, Partial, Virtual
Get Capability("Data Protection")
RAID, Erasure Coding etc.
Get Capability("Data Efficiency")
Dedupe, Compression etc.
Get Capability("Max num of subsystems")
Get Subsystem Capability()
"Number of Drives, Num of vols, number of n/w ports, NVMe capabilities"
Get Subsystems/NetworkPort Info
"Subsystem NQNs, Port Info"
Configure port QoS Settings (DCB)
Configure target switch port(s) QoS Settings (DCB)
Get Physical Drives()
Get Assignable Devices(NQN)
Register(host)
Configure NQN
Set Discovery NQN.Transport Addr
Configure n/w QoS Settings (DCB)
Configure host switch port(s) QoS Settings (DCB)
Compose
Create Assignable Device(physical drives, size, etc.)
Volume id
Configure NQN, Port
Assign Device to Host(Volume id, Tgt NQN, Tgt Ports, Host NQN, Rd/Wr permissions)
ConnectToTarget (Target NQN, host NQN, params))
Target related mgmt and config
Initiator related mgmt and config
Network related mgmt and config

12.7.7.1 Intel® RSD Manageability Model

The interface between the PODM and the PSMS will be based on Redfish/Swordfish with RSD-specific extensions.

The following Sections describe high-level manageability model based on *Redfish API Schema 2018.1*, (refer to Table 3), with RSD extensions to support NVMe-oF that uses Ethernet as Fabric type and RDMA as transport protocol. This model has been designed to be scalable so it supports pooled NVMe over PCIe, as well as other fabric types in the future.

12.7.8 NVMe-oF Fabric Model (Redfish Common Storage Fabric Model)

Figure 34 presents the Redfish common fabric model, which is the manageability model resource hierarchy for NVMe-oF. The model shown is not a complete list of Intel® RSD resources; it only shows objects directly related to NVMe-oF fabric modelling. It also uses the Redfish-defined advanced communication device object as the network interface in a system.

The Intel® RSD model shown in Figure 35 is based on the Redfish common fabric model and the Swordfish model (for volume management) but uses the simple Ethernet object as the representation of the network interface in a system, instead of the advanced communication device. The core of the common fabric model in RSD is the same as Redfish with a few OEM (RSD) extensions.

The details of the interface based on this model are described in the *Intel® RSD Storage Service API specification* and *Intel® PODM API specification* documents. Refer to Table 3.

Figure 34. Common Storage Fabric with Advanced Communication Devices

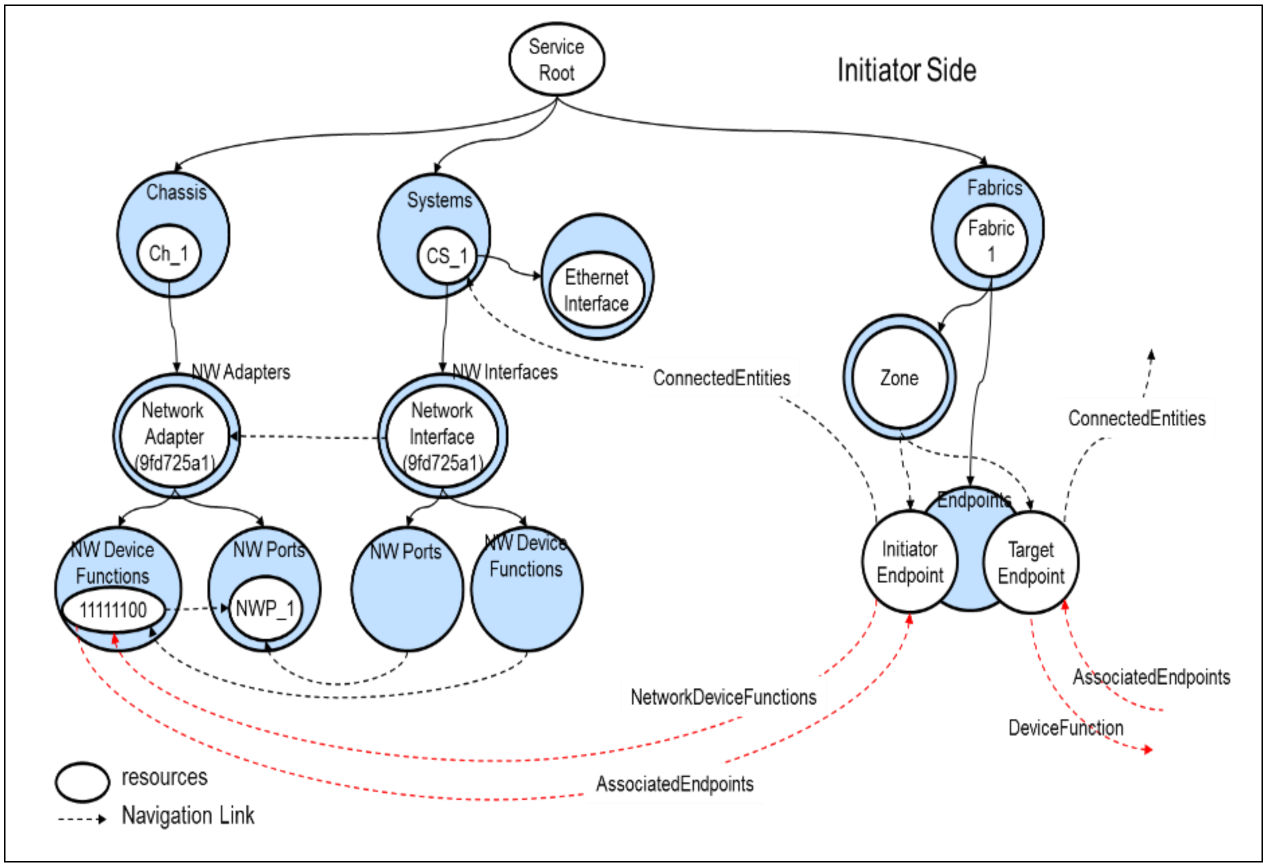


Figure 35. Intel® RSD Model for Common Fabric with Simple Ethernet Devices

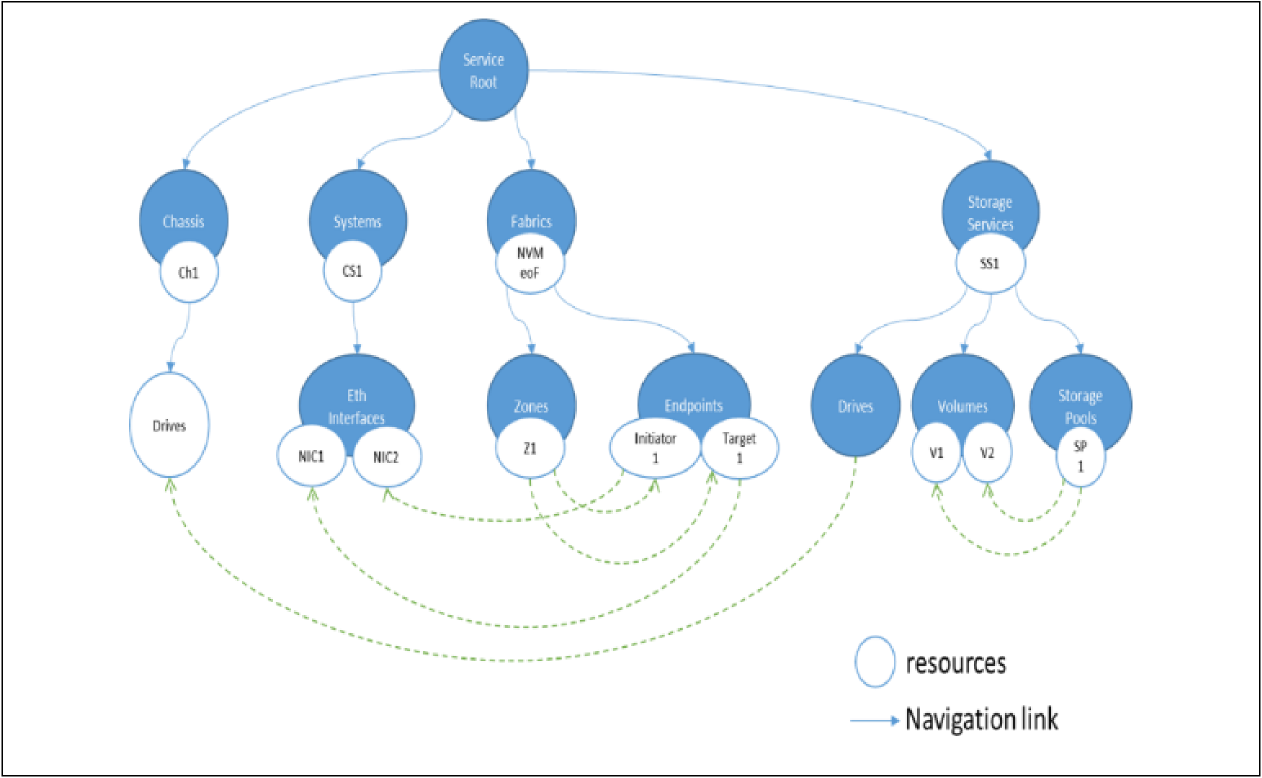


Table 14. NVMe-oF Fabric Model Resource Description

Resource Name	Resource Description
Computer System	A computer system represents a machine (physical or virtual) and the local resources such as memory, CPU, and other devices, which are accessed from that machine. In the NVMe-oF modeling this is a system, which initiates the NVMe-oF connection and RDMA transmission. A special type of computer system can also exist to model the system that hosts the storage subsystem.
Ethernet Interface	Resource provides detailed information about a network interface identified by NIC ID. In NVMe-oF modeling, it corresponds to network interface used for NVMe-oF communication, both from Initiator and Target perspective. The properties of this resource include MAC address and IP addresses used by the interface. For NVMe-oF modeling this resource also contain fully qualified domain name (FQDN).
Fabric	A fabric record represents a physical media layer, a medium used for communication between NVMe-oF Initiator and Target; it usually consists of a collection of Endpoints, a collection of Zones and possibly a collection of Switches of respective type. NVMe-oF only supports Ethernet as Fabric type.
Endpoint	A resource that represents the properties of an entity that sends or receives protocol defined messages over a transport. For NVMe-oF modeling, there are two types of Initiators used: Initiator Endpoints –sends or receives traffic from/to Computer System using Ethernet Interfaces associated with given Computer System, Ethernet as fabric type, and RDMA as protocol type. Target Endpoint – sends or receives traffic addressed to/from Storage Volume associated with pulled NVMe-oF storage (SSD Disk) using Ethernet Interface associated with given I/O System, Ethernet as fabric type, and RDMA as protocol type. These Endpoints also contain transport protocol properties (“`IPTransportDetails`”) used by the NVMe-oF Fabric Endpoints to communicate with each other. For current NVMe-oF definition, it specifies RDMA protocol and network address details.

Resource Name	Resource Description
Zone	A resource that represents a simple zone, which is an entity within specific Fabric, which group's number of Endpoints of the same Fabric Type. In NVMe-oF modeling, the Zone resource associates Initiator Endpoint to Target Endpoint, making possible to communicate via Ethernet using the protocol specified in the endpoint.
Storage	This resource defines a storage subsystem and its respective properties. A storage subsystem represents a collection of Volumes and a collection of Drives; it also keeps reference to Chassis hosting physical Drives. For NVMe-oF fabric modeling, the Storage is degenerated Redfish resource, since it lacks storage controllers, as they are not relevant here.
Volume	A resource used to represent a volume, virtual disk, logical disk, LUN, or other logical storage for a Redfish compliant implementation. It describes properties of a volume like Encryption used, block size of a volume, volume capacity, type of a volume (for example, raw device, Mirrored), a volume contains a link to a Drive resource which contains given Volume.
Drive	A resource used to represent a disk drive or other physical storage medium for a Redfish model. For NVMe-oF modeling, only supported Drive is NVMe based SSD.

12.7.8.1 Associating NVMe-oF Endpoints, Zones, and Volumes

An Endpoint is an instance of a network facing NVMe-oF initiator, or target subsystem, that participates in the NVMe-oF protocol to consume or deliver NVMe volumes. The Endpoint contains the necessary information regarding their view of Fabric and transport protocol used for NVMe-oF communication. The general case will have a Target EP instance for every subset of the volumes exported to initiators via the same subsystem. The Target EP contains links to the volumes it exports along with the access rights to these volumes.

For example, if an NVME-oF subsystem manages two volumes, V1 and V2, which are assigned to three Initiators I1, I2, and I3 as follows:

- I1->V1 (read/write access)
- I2->V2 (read/write access)
- I3->V2 (Read Only)

Then there would be three instances of the target Endpoint for the same subsystem (NQN):

T1(V1), T1'(V2), T1''(V2)

A zone instance represents the ability of an initiator to connect and access volumes (with specific rights) from a specific target subsystem. Extending the above example, three zones would be instantiated as follows:

- Z1->[I1, T1(V1)]
- Z2->[I2, T1'(V2)]
- Z3->[I3, T1''(V2)]

Note: The current Redfish model does not support the requirement for access permissions to be associated with the assignments, such that the same volume may be shared between multiple hosts with different access types (read vs. write).

§

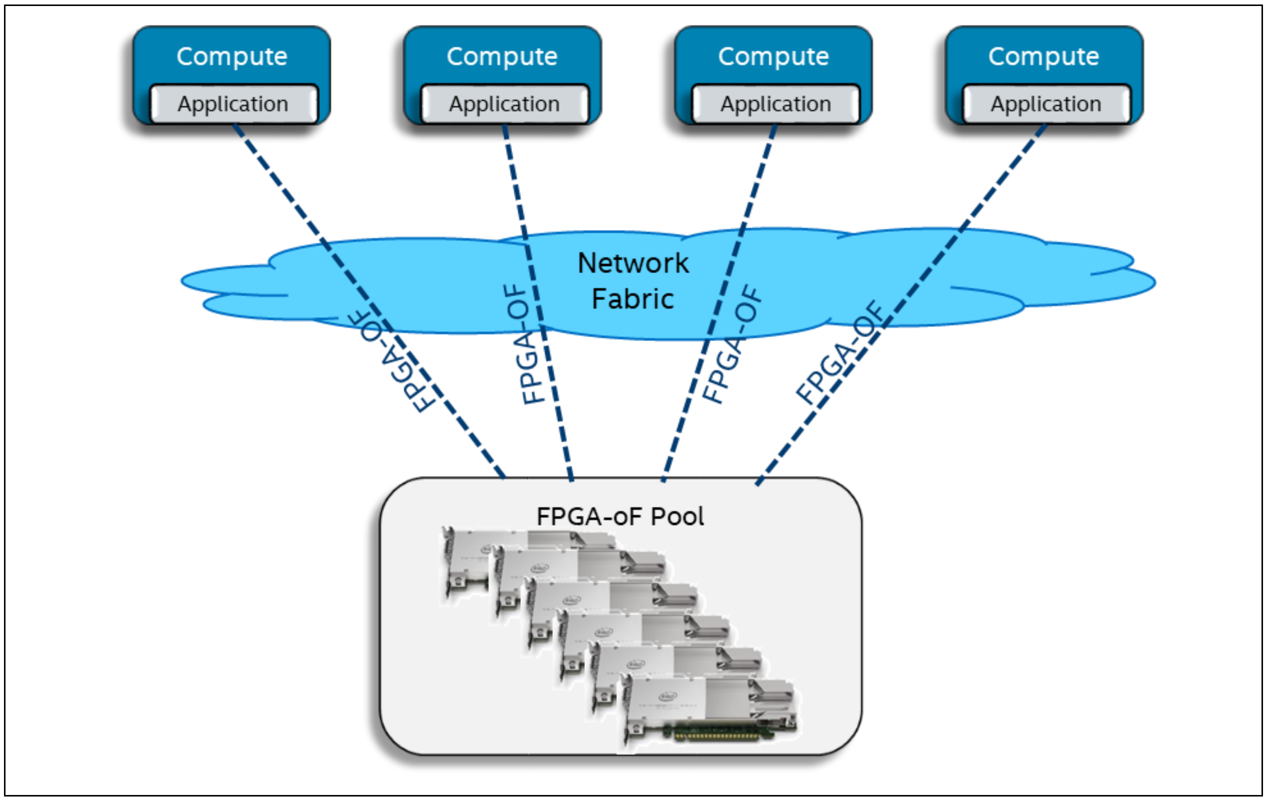
13.0 Ethernet Pooled FPGA

13.1 Overview

Pooling FPGAs over Ethernet fabric continues the RSD theme of disaggregating I/O devices from the compute node to maximize the flexibility and efficiency of composed systems. The Ethernet fabric provides a flexible high-radix connection environment to a pool of FPGAs, as shown in Figure 36.

The PODM can compose a Compute Node with FPGA accelerators that are logically attached across an Ethernet fabric using the FPGA-over-Fabric* (FPGA-oF*) transport protocol defined in the FPGA-oF Protocol Architecture Specification listed in Table 3. This has the advantage over the PCIe pooling described in Section 6.0, PCIe* Direct Attached Pooled I/O Design Guidelines that is constrained by the physical cable and attach points of the PCIe switch layer with a limited number of hosts connected to the FPGA PCIe pool.

Figure 36. FPGA Pooling over Ethernet



Major considerations with Pooled FPGAs are the bandwidth requirements of the anticipated workloads, the processing rate of the FPGA accelerator, and the capacity of the fabric connection to the pool of FPGAs. These factors should be considered carefully when sizing the number of FPGAs in the pool and network connectivity. Refer to Section 13.4.5, Network QoS Support for network QoS considerations.

Note: The characterization of workloads is not within the scope of this document.

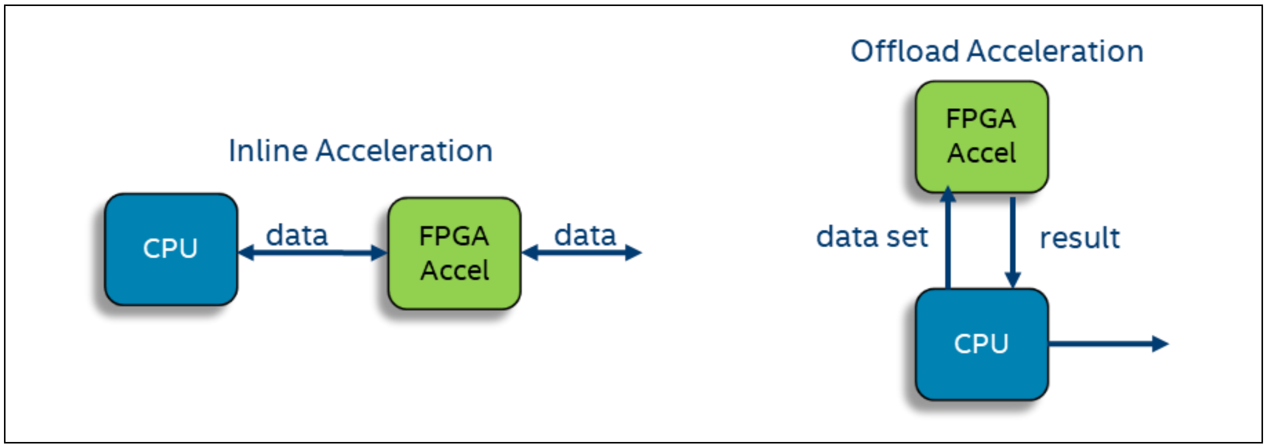
As far as use cases for FPGAs are concerned, there are two basic types of acceleration: inline acceleration and look aside (offload) type acceleration as seen in Figure 37. The inline or bump in the wire implies the acceleration is a serial operation within the data path. Such examples would be networking or storage acceleration for data encryption or compression.

Typically this type of acceleration has high bandwidth requirements and is not conducive to FPGA pooling over fabric as it would tend to consume the network connection to the pool. This type of acceleration workload would typically be better served with PCIe pooling described in Section 6.0.,PCIe* Direct Attached Pooled I/O Design Guidelines.

FPGA pooling over fabric may work well for the use case of look aside or offload type accelerations. In this use case a compute node sends a data set and then a command to perform the work on that data set to the accelerator and it will return a set of results as shown in Figure 37. Such an example would be imaged inferencing when an image is sent to the FPGA accelerator, and the accelerator performs object detection, returns object boundaries, and classifications.

The application network load for the remote FPGA depends on how many frames per second are being processed and the size of the frame.

Figure 37. Types of Accelerations



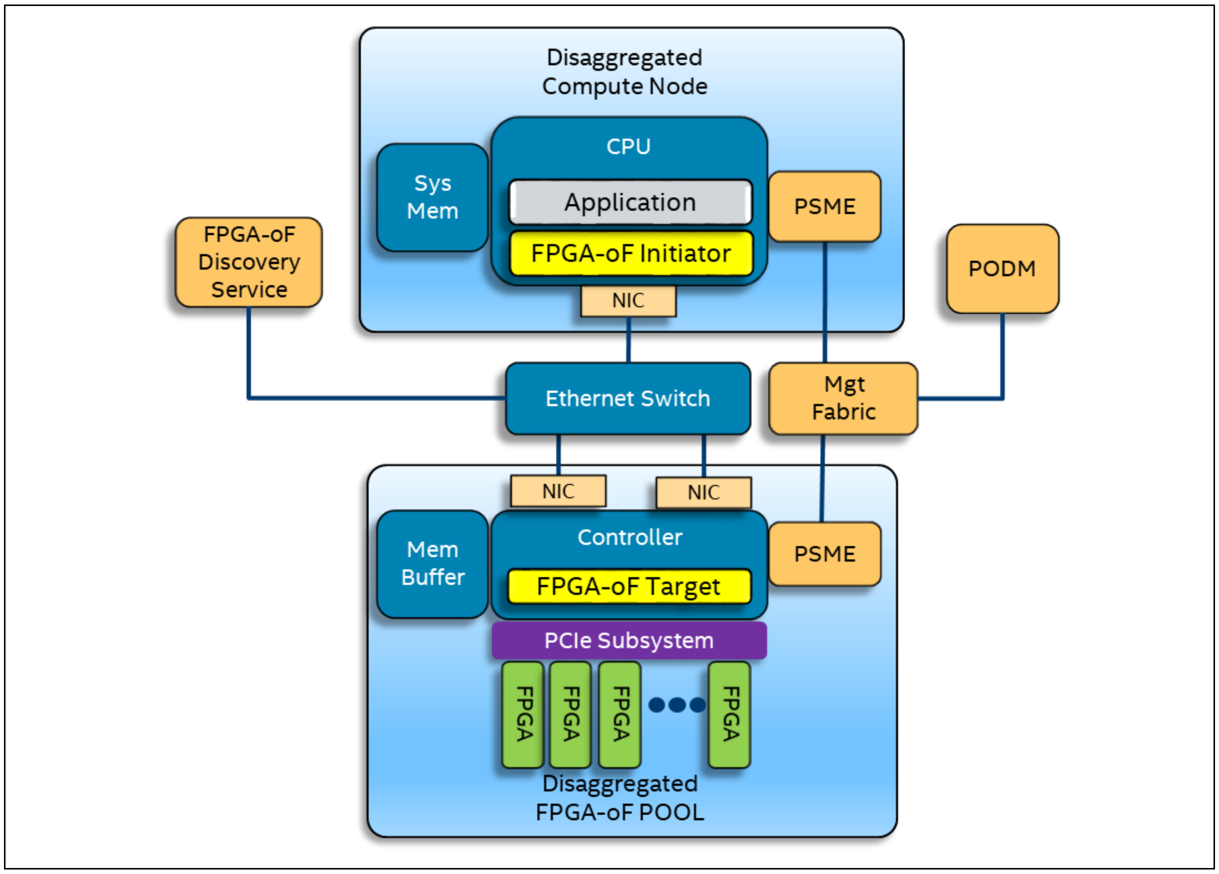
13.2 FPGA-oF Pooled Architecture

The Ethernet fabric connection of the FPGA is accomplished using a transport protocol defined in *FPGA-oF Protocol Architecture Specification* refer to Table 3, for details on the transport protocol. This section will provide an overview of the RSD Pooled FPGA-oF system architecture.

13.2.1 Pooled FPGA-oF Topology

The disaggregated compute node is known as the “initiator”. The disaggregated FPGA Target enclosure is known as a “Pooled FPGA-oF Target System.” The basic topology is shown in Figure 38.

Figure 38. Pooled FPGA-oF Basic Topology



The purpose of the pooled FPGA-oF Target Controller in Figure 38 is to perform the translation between the FPGA-oF commands over the transport to the PCIe* FPGAs. The controller can be implemented with CPUs or control logic implemented in a bridge device such as an FPGA or ASIC. There is typically buffer memory associated with the controller used in processing the FPGA-oF transport commands. FPGAs may also use this memory depending on the application usage.

The Controller interfaces to the network fabric via NICs. The discovery services will list the network protocols, i.e. RDMA, TCP and others that are configured and available on the pooled FPGA-oF target system for the initiator. It is possible for the controller to support a mix of protocols such as TCP from one host and RDMA from another. The Controller may support multiple network interfaces to provide greater bandwidth or multi-pathing.

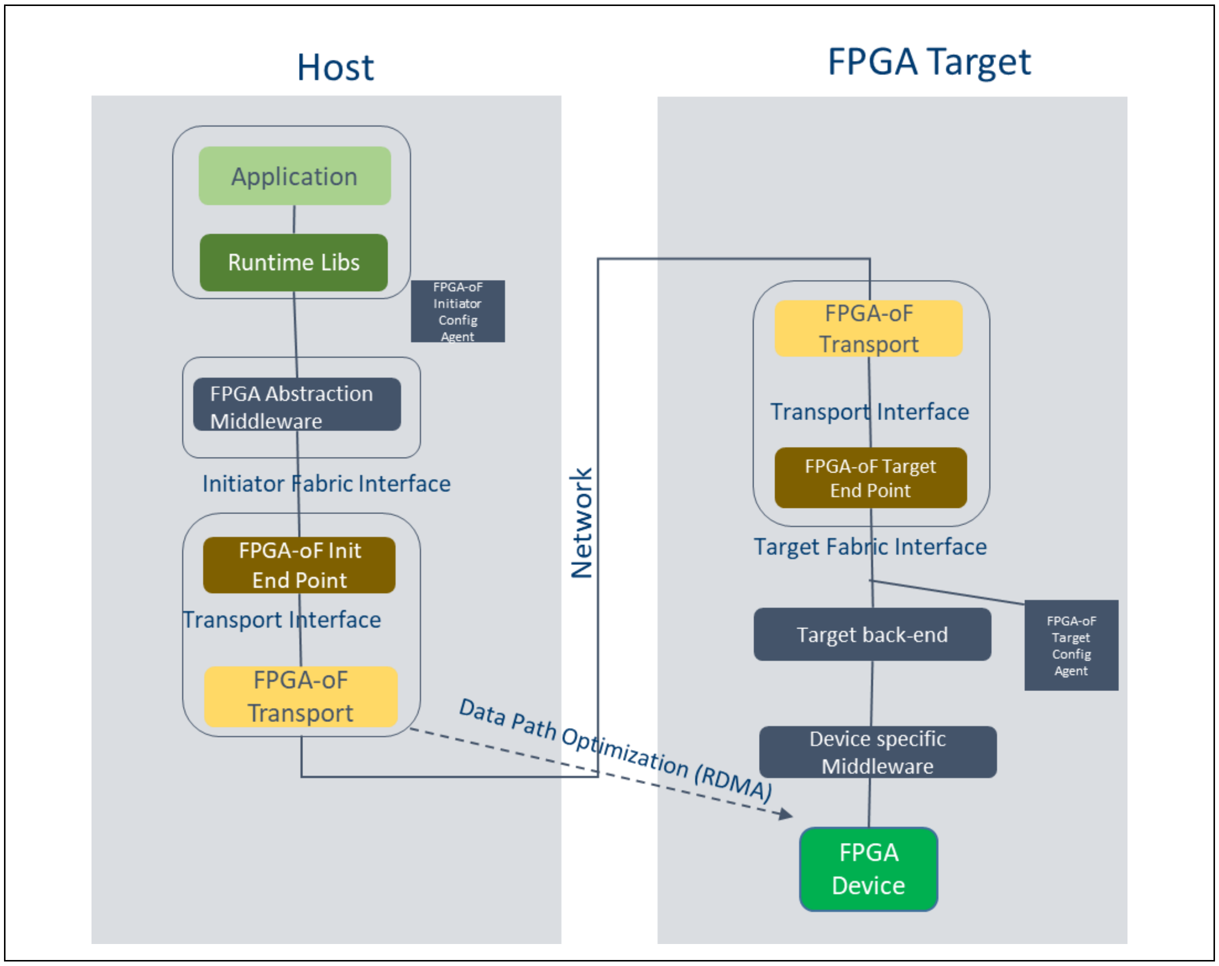
The PCIe subsystem is responsible for the electrical and logical connection of the FPGA PCIe interface to the Target Controller. The FPGA may be directly attached to the controller or through a PCIe switching layer to provide fan out. The PCIe switching layer is generally statically configured and not necessarily managed under RSD management. Refer to Section 6.0.,PCIe* Direct Attached Pooled I/O Design Guidelines in the case where it is RSD managed.

The FPGA modules are typically add-in cards having a PCIe-compliant interface. They may contain local memory and other I/O features that need to be exposed to the pooled FPGA-oF target system PSME. However, this document addresses only FPGA devices exposed on the PCIe interface of the FPGA module.

13.2.2 Framework of Components

Figure 39 shows the framework of components for the Initiator and the Pooled FPGA-oF target systems.

Figure 39. Framework of Components



The main components as shown in Figure 39 are:

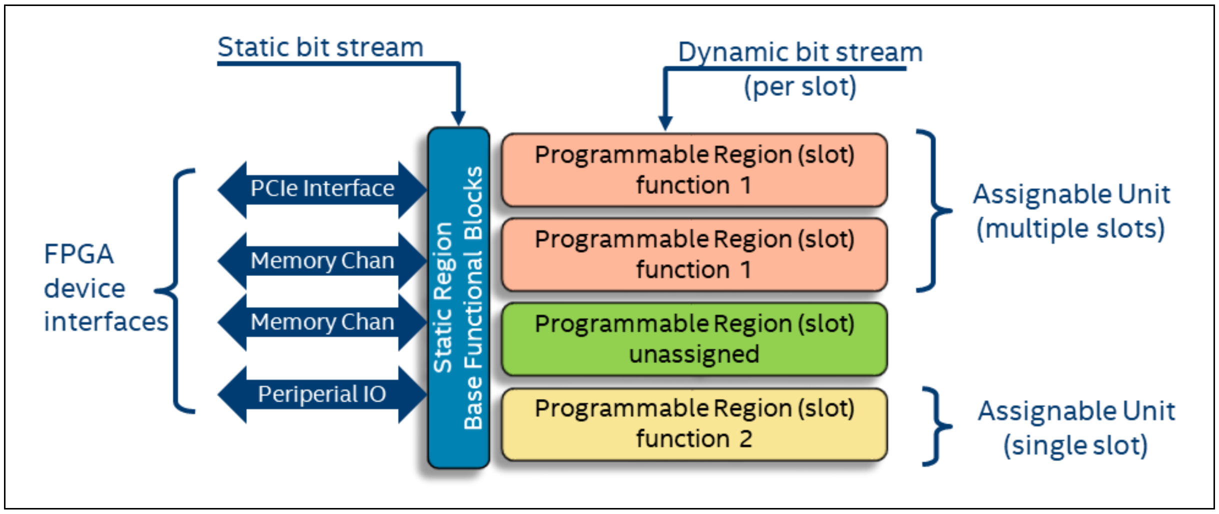
- Software components running on the Initiator (Host), namely the application and runtime layers, FPGA abstraction middleware and pooled FPGA abstraction software based on FPGA-oF.
- Components of the pooled FPGA-oF target system including the pooled FPGA abstraction layers based on FPGA-oF that handle the control and data plane from the initiator, pooled FPGA-oF target backend to drive the FPGA resource, FPGA device-specific middleware, and the physical devices.
- Management agents on the Initiator and pooled FPGA-oF target system to configure and manage the above components.

13.2.3 FPGA Reconfigurable Partitioning

Reconfigurable Partitioning provides a flexible means of programing or re-programming areas or regions of the FPGA without disrupting the service of an application that is running on other regions. This feature is supported by modern FPGA technology from multiple vendors and is supported in RSD Pooled FPGAs.

FPGAs that support this feature has two basic region types, static and dynamic (reconfigurable). Dynamic regions within the FPGA are referred to as programmable slots. If an FPGA device does not support partitioning, then a single bit stream is loaded to define the complete function(s) of the FPGA. The following terms are used in this document to the programing nature of the FPGA as shown in Figure 40.

Figure 40. FPGA Reconfigurable Partitioning


Static bit stream
Dynamic bit stream
(per slot)
FPGA device interfaces
PCIe Interface
Memory Chan
Memory Chan
Periperial IO
Static Region
Base Functional Blocks
Programmable Region (slot) function 1
Programmable Region (slot) function 1
Programmable Region (slot) unassigned
Programmable Region (slot) function 2
Assignable Unit (multiple slots)
Assignable Unit (single slot)

Static Region

The static region is the programmable area in an FPGA device that:

- Provides the basic functional blocks for the external interfaces of the FPGA device such as PCIe bus, memory channels and other I/O
- Makes the blocks available to programmable regions.

This area is not affected by loading dynamic bit streams into the programmable areas.

Programmable Region

A programmable region defines an area or slot within the FPGA for a usable, functional instance. Refer to the FPGA vendor specification for the definition, structure, and constraints of these programmable regions.

The management agent on the pooled FPGA-oF target system needs to be able to expose the availability of these slots such that it can convey to the PODM the ability to be utilized or loaded with a bit stream.

Static Bit Stream

The static bit stream, e.g., the blue bit stream for Intel® FPGAs, may be loaded into the static region of the FPGA at power-on or initialization providing basic PCIe interface along with and other static functional blocks. This allows the FPGA to receive further programming of the other programmable regions via the PCIe interface.

Dynamic Bit Stream

The dynamic bit stream is a partial bit stream, e.g., green bit stream for Intel® FPGAs, and provides the functional instance for the application that occupies a slot in the FPGA. As stated above, these programmable slots can be programmed and re-programmed without affecting or disrupting the service of the static or other programmable slots

Assignable Unit

The assignable unit is one or more programmed slots that constitute a useable function that can then be allocated or assigned to an initiator.

13.2.4 Remote FPGA Programming Model

Three basic FPGA device programming use cases are envisioned:

- **Firmware FPGA Programming:**

  In this use case, the function of the FPGA device is defined from its local firmware load upon initialization. The device is responsible for loading its bit stream functional blocks and presenting itself as a particular device type to the PSME management plane.

  The function remains persistent with the local firmware, and neither the PSME nor the Target controller may change its loaded function after initialization. Functional changes are only permitted via firmware updates in accordance with Intel® RSD specifications listed in Table 3. In this case, it is expected that the FPGA device function is fully operational at initialization time.

- **PODM Managed FPGA Programming:**

  In this use case, the PODM is solely responsible for managing the functional blocks for the FPGA. The baseline operational state may be loaded by its local firmware enabling its primary electrical interfaces such as PCIe, memory and I/O channels. The PODM is able to change or reprogram this baseline functionality, as well as add or remove other functional blocks through the OOB management channels.

  The persistence of the function instance is under the control of the PODM. In other words, the functional instance may remain across multiple composed assignments with other systems.

- **Initiator Managed FPGA Programming:**

  In this case, the Initiator Compute Module that is assigned an FPGA programmable region is able to load functional block bit streams to the device over the fabric using FPGA-oF. The FPGA device is expected to be initialized to a baseline operational state where at a minimum the PCIe interface to the pooled FPGA-oF target controller is functional for in band programing from the initiator via the FPGA-oF network connection.

  The PODM is responsible for returning the device to its original baseline operational state upon release of the device from the composed node, purging any functional blocks loaded by the Initiator Compute Module.

13.2.5 FPGA Virtualization and Sharing

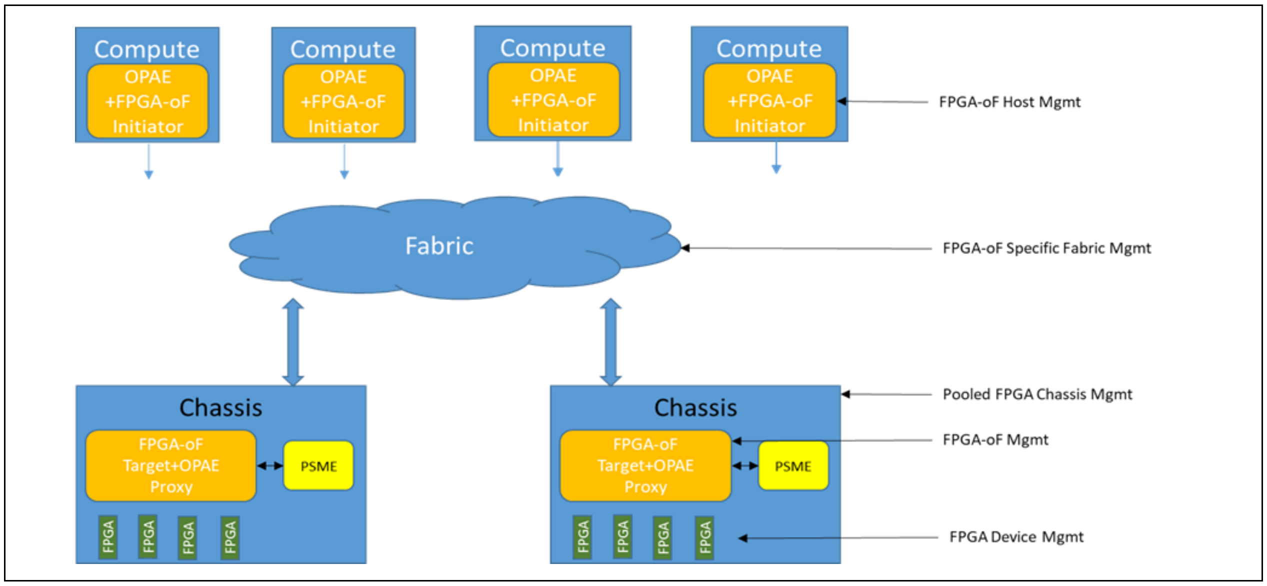
In the case of FPGA pooling over Fabric, virtualization of an FPGA to the initiator over the fabric is handled by the pooled FPGA control processor complex. This allows the pooled FPGA-oF target system to expose a programmed function or a programmable slot, as an assignable device, to an initiator.

The FPGA device can be shared by allocating the assigned units of the FPGA to various initiators. Sharing of an assignable unit with multiple initiators would require the cooperation of the application using the unit.

13.2.6 Management Model

To realize comprehensive management of FPGA pooling over Ethernet, the management model spans across several domains as shown in Figure 41.

Figure 41. Management Domains in an RSD Pooled FPGA Deployment



A typical pooled FPGA-oF target system is to consist of an enclosure or chassis housing the physical FPGA devices, the control processor complex needed to run the control plane for the FPGA pool, and a network interface able to access the pooled FPGA-oF target system from compute nodes over the network.

The exact number and type of FPGAs within a chassis may be implementation dependent. The physical chassis will be managed via the RSD Chassis management interface (part of Intel® RSD v1.2). Some implementations may integrate the processing elements and the network interface as a single hardware component, such as in the case of a pooled FPGA chassis with a hardware bridge that contains both a control processor complex and the network interface. Implementations may also have the physical FPGA devices attached to the control processor complex via a PCIe complex.

The PCIe complex may be either a static or a dynamic tree that supports pooling. If a PSME exists for the PCIe complex, then it can be managed via the RSD PCIe device management (part of Intel® RSD v2.x). The target backend exports the management interfaces to configure the pooled FPGA-oF target system and the FPGA-oF protocol functionality. This specification will focus on the functional requirements of this layer in support of the Intel® RSD usage model.

The host side element of the pooled FPGA system is also a critical element in managing pooling because it participates in connecting a compute node to an FPGA resource from the pooled FPGA-oF target system as part of a logical node composition. This specification will address the requirements of this layer to support the Intel® RSD usage model.

The network is a last key piece of the puzzle since it provides the physical connectivity that enables the FPGA-oF protocol to disaggregate FPGA accelerator devices. This specification will address the requirements of the fabric to enable efficient usage of a converged network for FPGA pooling.

13.2.6.1 Discovery

The discovery and mapping of hosts to FPGAs in a pooled FPGA deployment is done by overlaying a logical connection over the physical network topology. The logical connection between an initiator and a pooled FPGA-oF target system (target) is made by the initiator first initiating a transport-level connection with a pooled FPGA-oF target system using fabric-specific transport mechanisms (for example, RDMA send/receive Queue Pair establishment), and then subsequently initiating an FPGA-oF level connection to the target system using the FPGA-oF CONNECT command as described in the FPGA-oF protocol definition. Refer to the *FPGA over Fabric Protocol Architecture Specification* in Table 3.

The first step is for the initiator to discover all the pooled FPGA-oF target systems in the network it has access to. This can be done via an RSD-supported Discovery Service. This service has a similar access mechanism as other controllers in RSD (e.g., Discovery Service controller for NVMe-OF). The Discovery Log contains entries that specify information for the initiator to connect to a pooled FPGA-oF target system.

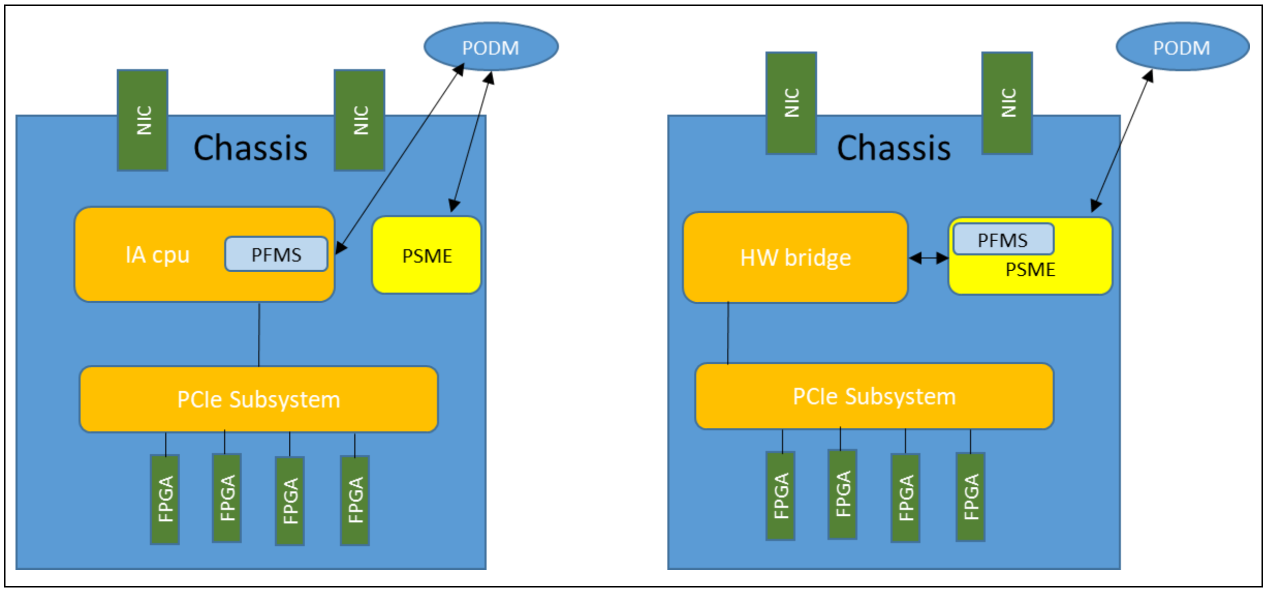
After the initiator system connects to the Discovery Service, the Discovery Log entries specify the fabric-specific addresses of the pooled FPGA-oF target systems assigned to that initiator. Each entry in the Discovery Log points to one pooled FPGA-oF target system. The actual list of entries returned to an initiator may vary and are specific to that initiator. The initiator is informed by the RSD management agent about the fabric-specific address of the discovery service to connect to.

If an RSD POD is introduced alongside a non-RSD POD that already has a discovery service, the RSD POD can either interface with the existing discovery service or provide its own discovery service for the resources and hosts within the RSD POD.

13.2.6.2 Logical

As mentioned in previous sections, a pooled FPGA deployment requires the participation of the Initiator, network, target system, and discovery service. Therefore, it follows that the Intel® RSD management model requires a management interface and functionality for each of these elements as shown in Figure 42.

Figure 42. Possible Realizations of Pooled FPGA-oF Target System Management



Each of the pooled FPGA functional elements will require an RSD agent that talks to the PODM to allow discovery and configuration of pooled FPGA-oF target systems and FPGA-oF services.

The interface to the PODM for these entities shall be an extension of the Redfish* model. Figure 42 shows two realizations of the pooled FPGA-oF target management entity based on the two common target implementations.

As shown, the x86 full stack target implementation will have a management agent running on the CPU of the target system. On the other hand, a hardware bridge-based implementation will have a control processor that can host this management agent. This specification refers to this management agent on the pooled FPGA-oF target system as the pooled FPGA Management Service (PFMS), regardless of the actual realization.

The PFMS, therefore, refers to a set of software components running in the control processor complex on the pooled FPGA-oF target system that allows RSD management (PODM) to enumerate, control and configure the pooled FPGA-oF target system.

As shown in Figure 42, in case of a software based pooled FPGA-oF target system implementation, the PFMS runs as a host agent on the pooled FPGA-oF target controller. In case of a hardware-bridge based controller, the PFMS functionality may be implemented as part of the pooled FPGA-oF target system PSME. Some of the functions of the PFMS include:

- Providing interfaces to allow the PODM to enumerate the physical FPGA modules and identify various properties of each FPGA module (e.g., number of programmable slots per module)
- Allowing PODM to reconfigure/program the programmable slots on a module
- Ensuring that the FPGA device and FPGA-oF stacks are correctly loaded and configured.
- Initializing and startup of the pooled FPGA-oF target stack and enabling it to start listening for client (initiator) connections
- Configuring the Access Controls (ACLs) on the pooled FPGA-oF target system to restrict initiator accesses to its corresponding assignable units

On the node hosting the discovery service, there is a management agent responsible for configuring the discovery service to provide automated set up of Initiator-Target System discovery/association.

A management agent on the initiator (Initiator FPGA-oF agent) is responsible for:

- Establishing a connection with the discovery service and discovering the pooled FPGA-oF target systems assigned to that initiator
- Configuring the initiator software stack with information about the discovered target systems
- Receiving and processing notifications from the discovery service to obtain information about changes to the assigned pooled FPGA-oF target systems

The Fabric (or Ethernet Network) does not require an agent specifically for FPGA-oF, but may be required to be configured to allow specialized treatment of data traffic to/from FPGAs with respect to priority, bandwidth, and congestion management.

13.3 Remote FPGA Functional Requirements

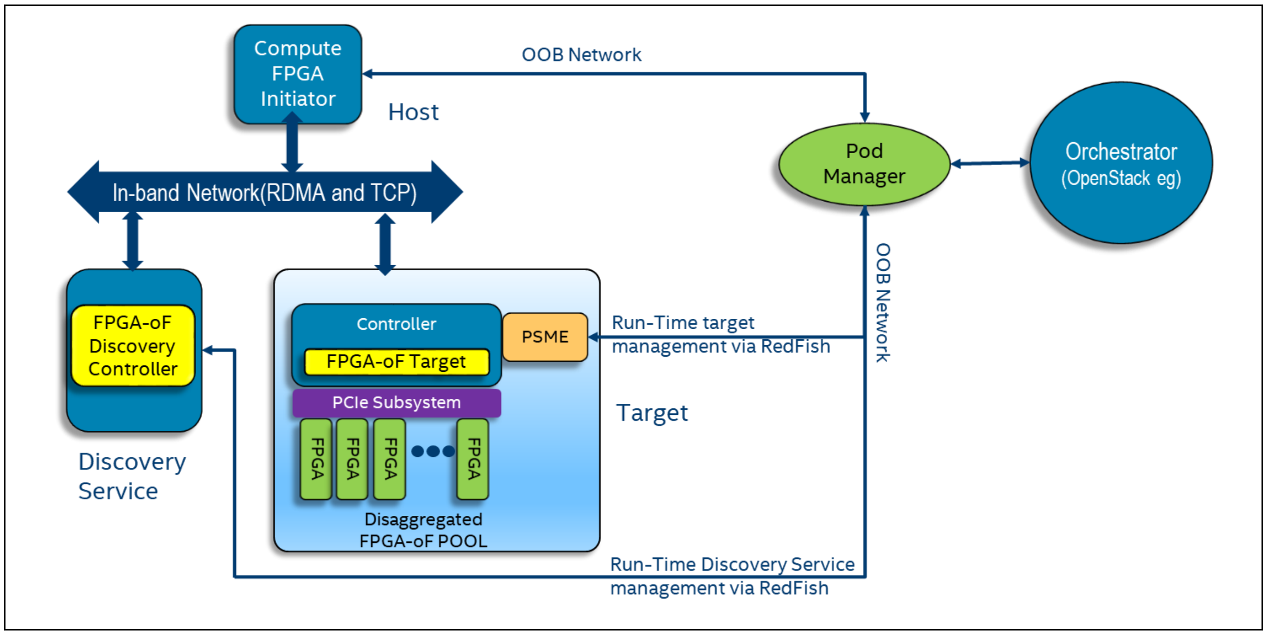
The various instantiations of the Pooled FPGA-oF Management Agents will deliver the Intel® RSD experience of automated management and dynamic composition of FPGA resources at the Rack level. Just to recap the management model described in Section 13.2.5, FPGA Virtualization and Sharing, the three types of elements that require management are at the Initiator, Pooled FPGA-oF target System, and network management for remote FPGAs. Each of these must provide a means of identifying the role they take in participating in RSD deployments as described below, namely:

- Pooled FPGA-oF target: This indicates that this system is a provider of FPGA devices
- Discovery: This indicates that this system is a provider of FPGA-oF Discovery Services
- Initiator: This indicates that this element is a host (Initiator) able to connect to Pooled FPGA-oF target systems and a discovery service.

13.3.1 Initiator (Host) and Target FPGA Composition

A key aspect of system composition is to be able to assign FPGA devices to hosts. To do that, all the pooled FPGA-oF target systems must be discovered, their connectivity information must be known, and a mechanism to associate them with hosts must be provided. With that in mind, all three FPGA-oF elements have functional requirements that need to be met.

Figure 43. Initiator, Target, and Discovery Management Roles


Compute FPGA Initiator
Host
OOB Network
Pod Manager
Orchestrator (OpenStack eg)
In-band Network(RDMA and TCP)
FPGA-oF Discovery Controller
Discovery Service
Controller
FPGA-oF Target
PSME
PCIe Subsystem
FPGA
FPGA
FPGA
FPGA
Disaggregated FPGA-oF POOL
Run-Time target management via RedFish
Target
OOB Network
Run-Time Discovery Service management via RedFish

13.3.1.1 Discovery Service Management

Optional:

The Discovery Service is the first stop for a host to connect to and find out about the target systems that contain FPGA devices it has access to. The PODM must know about the existence of a discovery service and the details necessary to configure it. Therefore, it follows that the discovery service must specify its role, network port information, and a Qualified Name to the PODM. It also needs to take FPGA-oF system and host information as input for configuring its Discovery Log Page for every host.

The Discovery service can run on any machine that has access to both the in-band and OOB network. It could be hosted alongside the PODM and supports both, TCP and RDMA protocols for the in-band network.

13.3.1.2 Target System Management

Required:

The pooled FPGA systems that provide FPGAs (target) contain properties that need to be discovered and configured. Such a system must advertise its role, its network port information, target ID, physical and virtual device information. The key function of the system is to selectively allow hosts to connect to it and have access to a set of FPGA programmable functions. This implies that it shall take the host and FPGA device information as input to create appropriate internal records that allow the host to access the logical FPGA programmable functions. Refer to Section 13.3.2.3.1, Authentication of Host and Target end points

The FPGA-oF protocol allows for the FPGA-oF Endpoints in the host and target to be able to authenticate each other to confirm communication is indeed with the desired, remote endpoint. To achieve this, it is necessary for the appropriate security credentials (e.g., public/private keys) to be configured at the host and target Endpoints.

Note: In a multi-tenant environment, the FPGA-oF Endpoints might reside within a tenant boundary (e.g., within a VM), and therefore, the authentication is for the Endpoint within that tenant. However as described above, It is assumed that the management SW will use the appropriate mechanisms to do this. Also, this needs to be done prior to establishing a connection between the host and the target.

The mechanisms to establish authentication between the host and target end points is described in Table 3, FPGA-over-Fabric Architecture Specification for more details.

13.3.1.2.1 Transport Channel Protection

Since the access to the device is over the network, transport level security is recommended to ensure confidentiality and integrity of the network channel. It is expected that the host and target will use IPSec or TLS on the network fabric for this purpose. It is expected that IPSec will be used in tunnel mode and the entire IP and Transport (TCP or UDP as the case may be) layers will be covered by the IPSec header.

Telemetry regarding FPGA parameters. The system may also support configuration of its parameters, such as target system ID via RSD as the mechanism to bootstrap it.

13.3.1.3 Initiator Management

Required:

Like the other two elements, the initiator shall also specify its role, its n/w port information, and initiator ID when discovered. The initiator ID can be assigned by the POD manager using the management interface. The Initiator must support at least one of the two mechanisms; out-of-band (OOB), defined in Section 13.3.1.3.1, OOB Initiator Configuration, or in-band (IB), defined in Section 13.3.1.3.2, In-band Initiator Configuration to configure the host.

13.3.1.3.1 OOB Initiator Configuration

As stated above, the initiator needs to know the identity of the Discovery Service (ID, transport address) and a method to force it to rescan for changes in the FPGA-oF system-wide configuration (such as new devices added/removed, targets added/removed, discovery service modified). The initiator shall also support configuration of its parameters such as Initiator ID by RSD as the bootstrap mechanism for the FPGA-oF Initiator. RSD will support an OOB mechanism to configure a host with the following:

- Initiator's assigned ID
- Discovery Service's IP address
- Discovery Service's Transport Protocol (`"TCP", "RoCE", "iWARP"`)
- Discovery Service ID

Optionally the configuration also includes the Discovery Service's public security key (for TLS communication between the Initiator and Discovery Controller).

The PODM will send this information to the host BMC/PSME, which will communicate with the RSD agent to configure the software stack with the information. The SW on the host can use the Redfish Host Interface to obtain this from the BMC/PSME. Alternatively, the BMC/PSME can configure BIOS variables with this information, and the SW can receive it from using BIOS variables.

Note: It is part of the implementation to define these variables as this is platform specific.

The Initiator will then get all its target information (including notification of events) from the discovery service, and no more OOB configuration will be required.

13.3.1.3.2 In-band Initiator Configuration

If a discovery service is being used and the OOB mechanism is not present (described in Section 13.3.1.3.1, OOB Initiator Configuration), the initiator may be provided information about the discovery service in one of the following ways:

- The in-band mechanism can be used to discover the discovery service on the network. This can be done by adding a DNS entry for the well-known Discovery service name in a local DNS server.
- Manual provisioning of the initiator with information about the Discovery Service

13.3.2 Security

Management of FPGAs within RSD has several security implications that must be addressed:

- Protecting programmable regions on the FPGA from unauthorized access
- Protecting the FPGA-oF Target system from unauthorized access from a compute host
- Protecting the Target system from unauthorized access via the Intel® RSD management function.

13.3.2.1 FPGA Assignable Unit Security

The loaded bit stream and any allocated memory on the physical FPGA must be secure from unauthorized accesses. There are two scenarios that need to be addressed for this case. Namely, data erase and protection during use. This revision of the spec only addresses the data erase scenario.

13.3.2.1.1 FPGA Programming Erase Support

Required:

One of the security issues in a dynamically composed multitenant system is that the same FPGA assignable unit may be successively owned by different applications/users. This requires the bit stream(s) of the assignable unit and any local memory that was allocated to the FPGA unit being unassigned from a host be erased before putting the programmable region(s) or slots back into the pool for subsequent assignment.

However, in the case of reassignment to the same application on a different host (due to workload migration), the data should not be erased. If the FPGA supports a static region for its local interfaces, (blue bits), it shall remain intact in the release or decomposition of the assignable unit to the initiator.

The PODM must inform the PFMS of the need to erase the assignable unit during an unassigned request. The default policy in the PODM shall be to erase the region during decomposition unless told to preserve it for reassignment.

The PFMS must erase all security keys associated with the released assigned unit of the FPGA as well during this process.

13.3.2.2 Pooled FPGA-oF Target PFMS Access Security

Required:

FPGA-oF enables an FPGA device to be visible and accessible on the network by a multitude of hosts and applications. This requires the target to have an interface that allows it to be provisioned with programmable assignable units, access control lists for hosts and so forth. This interface must be secured by authentication, confidentiality, and integrity mechanisms. This will follow the Intel® RSD security model where the PFMS must conform to any role-based security requirements of the PODM as well as be able to authenticate the PODM.

Access to pooled FPGA-oF target management APIs is only allowed with TLS. This means that the PODM and the PFMS on the pooled FPGA-oF target must establish a TLS connection before they can communicate. Establishing this TLS connection requires mutual authentication between the PODM and the selected FPGA-oF target. Once authentication succeeds, this TLS connection between the PODM and the PFMS will provide integrity and confidentiality.

Intel® RSD will support this mutual authentication between the PODM and PFMS by enabling provisioning of credentials to both the PODM and PFMS as described in the following subsections.

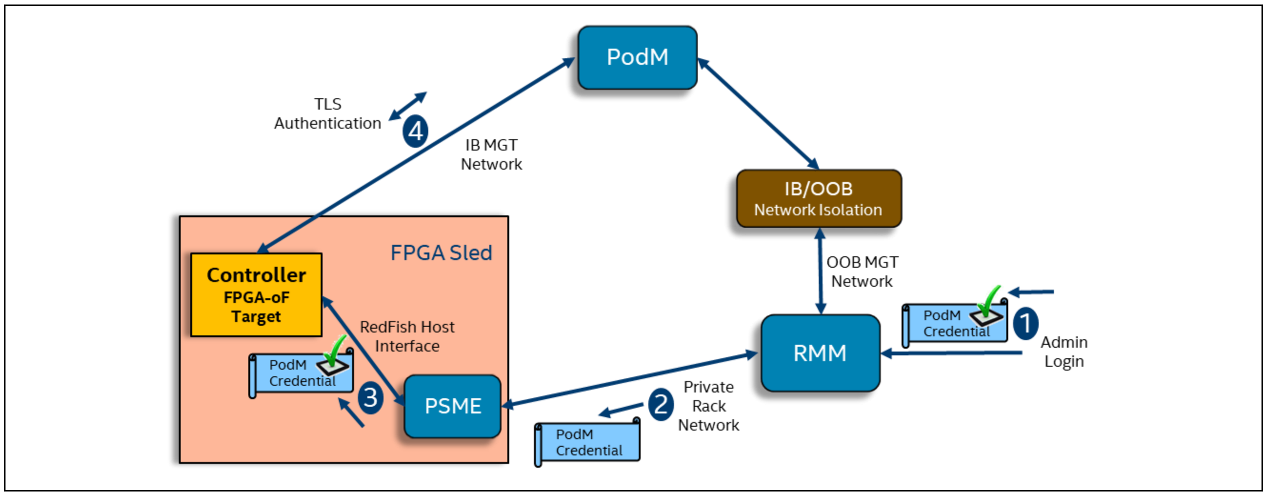
13.3.2.2.1 PODM Credential Provisioning to Target

1. The PODM authentication credential is provisioned to a rack (RMM) during initial rack set up.
2. During rack initialization, or when a new sled joins an already running rack, the RMM passes this credential to each sled's PSME using the rack private network.

3. The pooled FPGA-oF target system BMC can pass this credential to the target PFMS in one of the following ways:
   a. Wait for FPGA-oF target SW to use Redfish Host Interface functionality to retrieve PODM credential.
   b. Place this credential in a BIOS location where FPGA-oF target SW can retrieve it (using a BIOS call) once it is up and running.
   c. Manually configure PODM credentials in the target PFMS.
4. The pooled FPGA-oF target PFMS uses this credential to authenticate PODM as part of the TLS handshake.

Figure 44 illustrates the provisioning steps listed above.

Figure 44. Provisioning of PODM Credential to Pooled FPGA-oF Target System



13.3.2.2.2 Pooled FPGA-oF Target Credential Provisioning to PODM

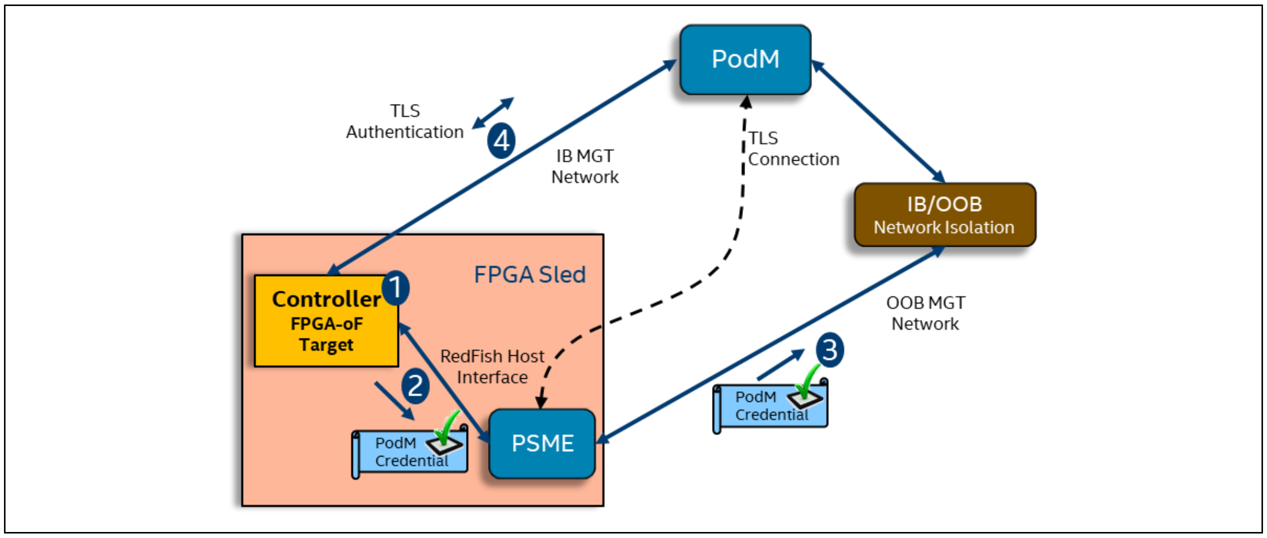
Figure 45 illustrates how to pass a public key credential to the PODM:

1. PFMS of the pooled FPGA-oF target system generates private/public key pair that will be used for authentication.
2. The public key credential is passed to the target system BMC using Redfish Host Interface functionality.
3. The PSME passes the public key credential to the PODM OOB using the OOB management network and secure connections created by the RSD management infrastructure (i.e., PSME to PODM TLS connection).
4. As an alternative to 2 and 3, the target PFMS certificate is signed using a CA certificate, and PODM verifies it against that certificate.

   The PODM and PFMS can now communicate securely using TLS.

The PODM will use this credential to authenticate the PFMS of the pooled FPGA-oF target system as part of the TLS mutual authentication. Illustrates the steps listed above.

Figure 45. Provisioning of FPGA-oF Target Credential to PODM



13.3.2.3 FPGA-oF Initiator Access Security

Recommended:

The FPGA over Fabric Transport protocol spec supports two levels of initiator and target secure communication, namely transport channel (network) protection on the wire and in band authentication. Refer to Table 3, FPGA-of-Fabric Protocol Architecture Specification, Security section of the FPGA-oF for more details.

It is expected that authorization is implemented by management software and complies with the particular Data Center management practices. The management of access control or authorization to the FPGA resources is beyond the scope of this specification.

13.3.2.3.1 Authentication of Host and Target end points

The FPGA-oF protocol allows for the FPGA-oF Endpoints in the host and target to be able to authenticate each other to confirm communication is indeed with the desired, remote endpoint. To achieve this, it is necessary for the appropriate security credentials (e.g., public/private keys) to be configured at the host and target Endpoints.

Note: In a multi-tenant environment, the FPGA-oF Endpoints might reside within a tenant boundary (e.g., within a VM), and therefore, the authentication is for the Endpoint within that tenant. However as described above, It is assumed that the management SW will use the appropriate mechanisms to do this. Also, this needs to be done prior to establishing a connection between the host and the target.

The mechanisms to establish authentication between the host and target end points is described in Table 3, FPGA-over-Fabric Architecture Specification for more details.

13.3.2.3.2 Transport Channel Protection

Since the access to the device is over the network, transport level security is recommended to ensure confidentiality and integrity of the network channel. It is expected that the host and target will use IPSec or TLS on the network fabric for this purpose. It is expected that IPSec will be used in tunnel mode and the entire IP and Transport (TCP or UDP as the case may be) layers will be covered by the IPSec header.

13.3.3 Telemetry

The physical components (CPU, network ports, and so on) in a target system should support the Redfish OOB telemetry interface defined for those components. In addition, the target should export telemetry on a per FPGA system.

13.3.3.1 Pooled FPGA-oF target system capabilities

The Pooled FPGA-oF target System shall report the following system level parameters:

Required:

- Number and type of FPGA modules populated in the target system.
- System level power and thermals

Optional:

- Utilization of Pooled FPGA-oF target control processor or bridge
- Power and thermal metric for each FPGA module

13.3.3.2 FPGA Device Telemetry

Required:

At this time there is neither a standard for FPGA metrics nor is access of metrics available across FPGA vendors and device types. However, the minimum required metrics for each FPGA device are as follows:

- FPGA device temperature, metric may be vendor specific.
- FPGA device power level, by programmable region, if available.

Optional:

- FPGA local memory bandwidth and utilization if present.
- Utilization of programmed Assignable function in the FPGA, this may be implementation specific.
- Programmed function performance metrics; this may be implementation specific.
- FPGA PCIe bandwidth utilization.

13.3.4 FPGA Target System Chassis Management

The FPGA target system chassis can be managed by both in-band (PFMS) and OOB (PSME) mechanisms. The management functions are accomplished via an SMBus structure connected to the management controller.

Required:

- Each FPGA module shall have a functional SMbus connection at system power on or system reset.
- Detection of FPGA Module presence in the Pooled FPGA-oF target system.
- Issue a reset to an individual FPGA Module. If this reset does not clear the FPGA programmable regions of the FPGA, then there must be a means to programmatically clear all programmable regions within the FPGA.
- FPGA-oF Target System power and thermal monitoring.

Optional:

- Issue, a Target Controller level, reset without affecting the loaded assignable units of the FPGA.
- FPGA module power and thermal monitoring.
- FPGA Module Firmware management and update.
- FPGA local resource management i.e. local memory.
- Power cycle individual FPGA module within Pooled FPFA-oF target system.

13.3.5 Hot-Plug

Optional:

There are two cases where hot-plug is relevant in an FPGA-oF deployment.

Add-Removal at the Host

FPGA devices in the initiator may be dynamically added and removed while the OS is running (equivalent to hot add/remove). The actual hot-plug of a physical FPGA module will require PCIe hot-plug support from the target platform.

Hot-Plug at the Target

If a target platform supports physical FPGA device hot-plug, then the PODM must be notified by the target PFMS of the arrival/departure of the devices. The PFMS can use the existing RSD state change notification API to notify the PODM of a state change. The PODM must then query the PFMS and determine the addition/loss of FPGA devices in the system.

Note: The requirements and mechanism used by a target to support FPGA module, hot-plug is implementation-specific and is beyond the scope of this specification.

13.4 Intel® RSD Management Software

13.4.1 Software Architecture

As mentioned above, the main components of an Intel® RSD compliant pooled FPGA Management Software Architecture are:

- Target system PFMS
- Initiator node FPGA-oF agent
- Discovery service

The software components needed for creating an Intel® RSD compliant pooled FPGA deployment are shown in Figure 46. The general implementation of the pooled FPGA-oF target or initiator is in a processor complex, running as part of an application middleware layer. Some implementations may have part of the target or initiator implemented in hardware for performance reasons. In either case, there is a control processor complex, each running the initiator and pooled FPGA management functionality.

The control processor complex is depicted in Figure 46 for both the host and the target, running the FPGA-oF software. The pooled FPGA Management Service (Target PFMS) runs on the same control processor complex as the pooled FPGA-oF target system, and interfaces with it to get and set the relevant information needed to manage it. This Target PFMS is an Intel® RSD component. It is expected that this control complex does not have direct access to the private OOB management network. Refer to Section 13.2.2, Pooled FPGA-oF Target PFMS Access Security to see methods of establishing a connection to the PODM.

The initiator-side FPGA-oF agent implements the management system to perform tasks such as:

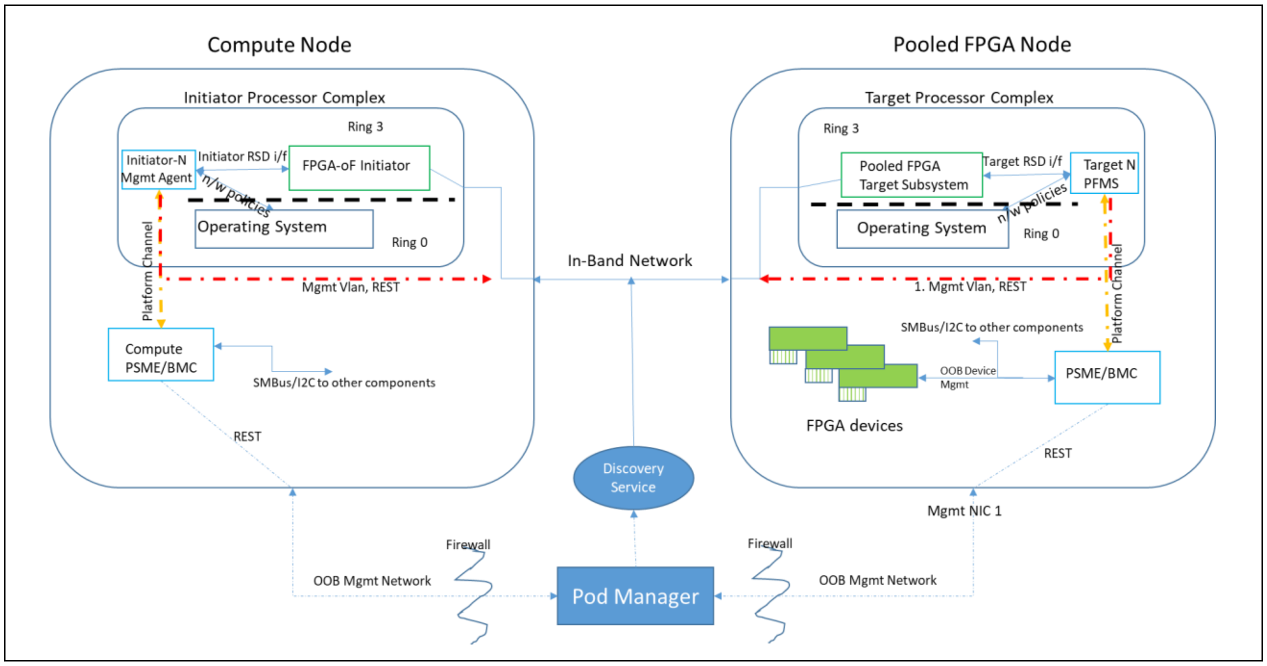
- Connecting to the Discovery service to discover the pooled FPGA-oF target-systems provisioned for that initiator
- Configuring the initiator with the information needed to access the pooled FPGA-oF target subsystem

Once the Initiator FPGA-oF agent configures the software stack on the initiator with its security credentials necessary to allow it to establish connections with pooled FPGA-oF targets, the initiator can then communicate with the pooled FPGA-oF target system directly. The initiator FPGA-oF agent requires information from the PODM, which can be done by communicating either directly or indirectly through the compute PSME/BMC.

The indirect communication can be done using the compute PSME/BMC to communicate with the initiator FPGA-oF agent via BIOS runtime variables (similar to Section 12.6.3.3.1, OOB Host Configuration with Discovery Service).

Direct communication requires the initiator FPGA-oF agent to communicate with the PSME/BMC via the in-band network for host configuration, by using the Redfish Host Interface (similar to Section 12.6.3.3.3, I/B Host Configuration).

Figure 46. Intel® RSD FPGA Pooling Software Components



13.4.2 Pooled FPGA-oF Target Management Interface Functional Requirements

Required:

This section lists the information exposed by the Pooled FPGA-oF target system to the PODM to deliver the RSD management requirements. Some of the functional elements are optional because they are dependent on the level of functionality of the Pooled FPGA-oF target. The actual realization of the interface between the PODM and the Pooled FPGA-oF target system is implementation-dependent and can be a combination of configuration files, CLI and/or IPC.

Table 15. Pooled FPGA-oF Target Management Interface Functional Requirements

Function	Input	Output	Required/ Optional	Comments
Configure Service Role	Role type: FPGA Accelerator Pool	None	Required	Provider of Data Service
Identify FPGA Devices	None	List of FPGA devices	Required	Returns list of structures with each entry describing one physical FPGA device including FPGA type, model, number of slots, PCIe Segment/Bus/Device/Fn.
Identify Assignable Units	Device ID	List of Assignable units	Required	Returns list of assignable unit IDs corresponding to the given physical device ID.

Function	Input	Output	Required/ Optional	Comments
Configure the Assignable Unit property	Device ID, Assignable Unit ID	None	Required	The PODM or Orchestration layer is responsible for assigning the Unique ID for the assignable unit.
Identify Network Ports	None	List of Port IDs	Required	PFMS generates unique ids for all the in-band data plane Ethernet ports
Identify Network Port Info	Port ID	Transport Address, Type, Service Identifier	Required	Address: String Type: "`RoCE", "iWARP", "TCP".`
Identify Capability	Capability Type: Virtualization	None, Full Device, Partial Device (Slot), Virtual Device	Required	**None** implies that the target system does not support virtualization **Full Device** implies the system can export a full FPGA device. **Partial Device implies** it can export individual slots of an FPGA device. **Virtual Namespace** implies it can export a device that is abstracted from the actual physical device.
Identify Capability	Capability Type: Authentication Level: Transport, FPGA-oF Endpoint	Fabric Secure Channel (IPSec, TLS, Other), FPGA-oF Endpoint in-band, None	Required	-
Identify Capability	Capability Type: Max limits	Max number of physical devices, max number of programmable slots, number of n/w ports	Required	This exposes the maximum capabilities of a target system with respect to the number of devices, slots, ports it can support. A return of 0 for any of these values implies no limit. This is based on the target system's design limitations due to say the amount of RAM in the sled. RSD can use this to ensure correct provisioning of the system.
Identify Properties	Security Credentials	Security Credentials info	Required	Security Credentials is the public key used by the system for transport channel secure communication.
Identify Device Properties	Device ID	Device Param Structure	Optional	Structure describing various FPGA device attributes including Static Bitstream ID, Number of ALMs, Memory (DRAM) capacity, Number of programmable slots, Name, and ID of the dynamic bitstream for each assignable device
Identify Statistics	None	Connected Initiator list	Required	-
Configure ID	UUID	None	Required	This pooled FPGA-oF target system will now be identifiable with this UUID

Function	Input	Output	Required/ Optional	Comments
Configure Network Ports	List of n/w ports, Transport addresses	None	Optional	This is required for systems that do not auto discover and auto-configure network ports.
Configure Network Port ID	n/w port, Pooled FPGA-oF target ID	None	Required	A 16-bit Port ID shall be generated unique to a pooled FPGA-oF target (a simple counter will work)
Assign Programmable Resource to Initiator	Target ID, Initiator ID, FPGA Device Physical ID, Assigned Resource ID, Allowed Accesses	None	Required	Allowed Accesses - Reset, Reconfigure, None Assigned Resource ID of 0 implies the physical device is assigned.
Logs	Target, Device ID	Log Content	Optional	Log content is relevant to the specific implementation
State Change Notification	None	None	Required	This allows the target to notify the PODM about changes in its state.

13.4.3 Pooled FPGA Discovery Service Management Interface Functional Requirements

Required:

The Discovery Service is a service that provides information about pooled FPGA-oF target systems to the initiator nodes. This can optionally be implemented as part of the PODM. Table 16 lists the required interfaces for a system implementing a Discovery Service.

Table 16. Pooled FPGA Discovery Service Interface Requirement

Function	Input	Output	Required/Optional	Comments
Configure Service Role	Discovery	None	Optional	Provider of Discovery Service
Identify Ports	None	List of Port IDs	Optional	–
Identify Port Info	Port ID	Transport Address, Type, Service Identifier	Optional	Address: String Type: "`RoCE`", "`iWARP`", "`TCP`".
Identify Properties	UUID	Port IDs, Security Credentials	Optional	Security Credential is the public key used for transport channel secure communication
Configure ID	UUID	None	Required	Configure the UUID for the Discovery Service
Configure Network Ports	List of n/w ports, Transport addresses	None	Optional	This is required for Discovery systems that do not auto discover and auto-configure network ports.
Configure Network Port ID	n/w port, Pooled FPGA Port ID	None	Optional	A 16-bit Port ID shall be generated uniquely for the Discovery Service (a simple counter will work)
Extract Discovery Log	Host UUID	List of Discovery Entries	Required	A Discovery Entry shall contain a Target UUID, Transport Address (IPv4 and v6), Transport Service ID, Port ID of the System, System Public Key

Function	Input	Output	Required/Optional	Comments
Configure Discovery Entry	Host UUID, Discovery Entry	None	Required	This is used by the PODM to set the discovery log entry for a host that needs to be associated with a target due to a composition request. Same Discovery Entry content as above.

13.4.4 Initiator Management Interface Functional Requirements

Required:

The primary requirement on the host side is to configure the initiator with the location of the Discovery Service at which point it is enabled to discover all the pooled FPGA-oF target systems available to it. If a Discovery Service is not available, then RSD needs to explicitly configure the host for every target. The following table lists the Intel® RSD management interfaces an Initiator needs to support.

Table 17. Pooled FPGA Host Interface Requirements

Function	Input	Output	Required/Optional	Comments
Configure Service Role	None	Host	Required	Initiator
Read Host ID	None	List of UUIDs	Required	Should be a list of 1 in the general case
Configure Host ID	UUID	None	Required	RSD shall generate the host UUID using the UUID of the Initiator PSME
Configure Discovery Info	Discovery Service Info	None	Required	The input will contain the Discovery Service UUID, Transport Address, Transport Protocol Type (RDMA, TCP and so on) and Transport Service ID (Port number) of the Discovery Service
Discovery Log Change Notification	Discovery Service Info	None	Optional (if Discovery Service does not support in-band notification)	Indicates to the host that a discovery log state change has occurred
Connect To Target	Target UUID, Target Transport Info (protocol type, port), and flags	Success/ Failure	Optional (if Discovery Service is not present/used)	Indicates to the Initiator that it should connect to a pooled FPGA-oF target and enumerate devices.

13.4.5 Network QoS Support for FPGA-oF

Optional:

There are three elements to manage when it comes to delivering network QoS for Pooled FPGA. All three entities require appropriate interfaces to allow configuration of QoS policies.

13.4.5.1 Host and Target

Set Network Port DCB Settings by doing the following:

- Enable DCB on the NIC.
- Configure PFC for Pooled FPGA TC.
- Configure priority to traffic map for Pooled FPGA traffic (using port number).
- Configure BW percentage for Pooled FPGA TC if desired.

13.4.5.2 Switch

Set Switch DCB settings by doing the following:

- Configure PFC for the Pooled FPGA traffic class.
- Configure bandwidth percentage on priority group that includes the Pooled FPGA traffic class.
- Enable DCBx.

13.4.6 Error Handling for FPGA over Fabric

There are two categories of errors that RSD will need to manage:

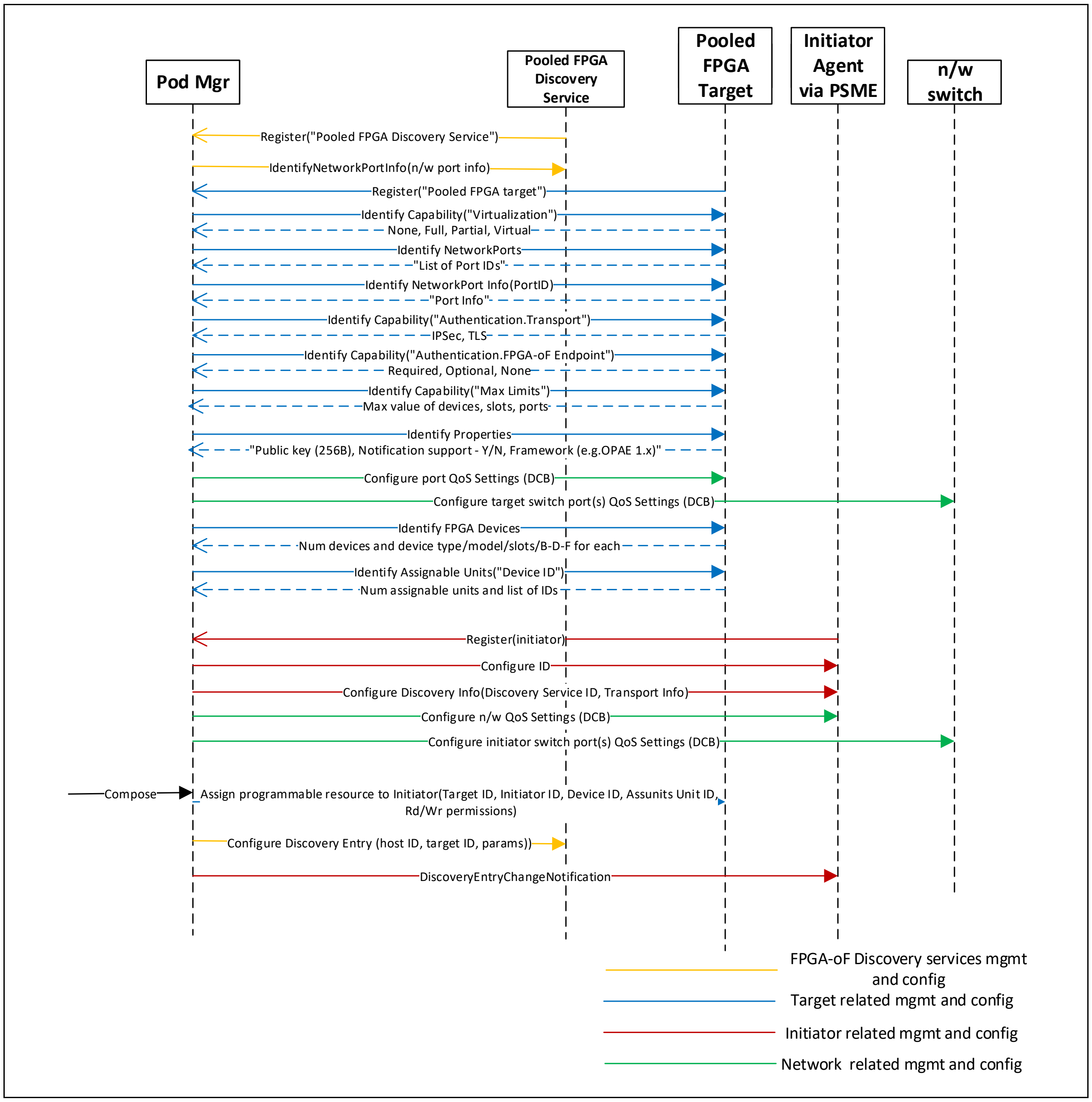
- (**Required**) RSD Operations Errors: These are errors encountered during operations, such as Composition/Decomposition. These should be handled using existing RSD mechanisms.
- (**Optional**) Pooled FPGA Errors: FPGA-oF specification specifies a set of error conditions whose status would be useful to capture and report. These errors can be reported via a LogService collection resource. Some specific errors in this category could be:
  - Connection errors – Report reasons such as transport failures
  - Error status of fabric completion commands (as specified in the *FPGA-over-Fabric Protocol Specification*, refer to Table 3)

13.4.7 Common Configuration Flows for FPGA over Fabric

Figure 47 shows example functional flows required to configure initiators and targets as well as a discovery service (if it exists).

Note: For security reasons, the initiator is not permitted to communicate directly to the PODM. Therefore the initiator must use the PSME as a proxy to communicate to the PODM. For clarification, Figure 47 labels this as "Initiator Agent via PSME".

Figure 47. Management and Configuration Flows


Pod Mgr
Pooled FPGA Discovery Service
Pooled FPGA Target
Initiator Agent via PSME
n/w switch
Register("Pooled FPGA Discovery Service")
IdentifyNetworkPortInfo(n/w port info)
Register("Pooled FPGA target")
Identify Capability("Virtualization")
None, Full, Partial, Virtual
Identify NetworkPorts
"List of Port IDs"
Identify NetworkPort Info(PortID)
"Port Info"
Identify Capability("Authentication.Transport")
IPSec, TLS
Identify Capability("Authentication.FPGA-oF Endpoint")
Required, Optional, None
Identify Capability("Max Limits")
Max value of devices, slots, ports
Identify Properties
"Public key (256B), Notification support - Y/N, Framework (e.g.OPAE 1.x)"
Configure port QoS Settings (DCB)
Configure target switch port(s) QoS Settings (DCB)
Identify FPGA Devices
Num devices and device type/model/slots/B-D-F for each
Identify Assignable Units("Device ID")
Num assignable units and list of IDs
Register(initiator)
Configure ID
Configure Discovery Info(Discovery Service ID, Transport Info)
Configure n/w QoS Settings (DCB)
Configure initiator switch port(s) QoS Settings (DCB)
Compose
Assign programmable resource to Initiator(Target ID, Initiator ID, Device ID, Assunits Unit ID, Rd/Wr permissions)
Configure Discovery Entry (host ID, target ID, params))
DiscoveryEntryChangeNotification
FPGA-oF Discovery services mgmt and config
Target related mgmt and config
Initiator related mgmt and config
Network related mgmt and config

§