**Open Fabrics Management Framework**

**Document Outline**

Table of Contents

[2 Overview 3](#_Toc115068512)

[3 Architecture of the OpenFabrics Management Framework 3](#_Toc115068513)

[4 OFM Fabric Agents 5](#_Toc115068514)

[5 Resources Configuration 5](#_Toc115068515)

[6 Fabric Configuration 5](#_Toc115068516)

[7 Authentication and access control 5](#_Toc115068517)

[8 Systems composability 6](#_Toc115068518)

[9 Events and logs 6](#_Toc115068519)

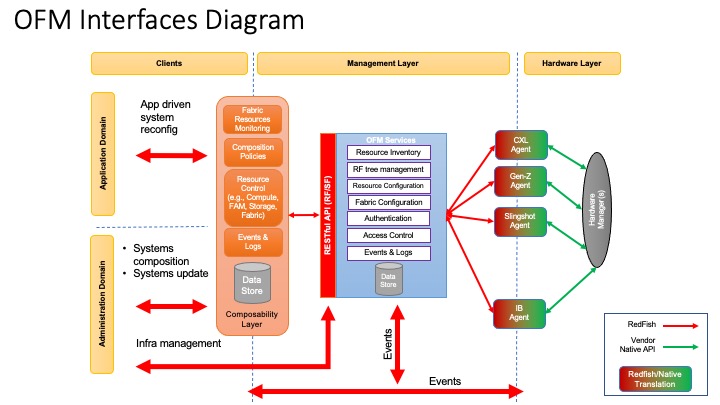
[10 Security 6](#_Toc115068520)

[11 Use cases 7](#_Toc115068521)

# Overview

* Provide universal orchestration and management tools for computational resources and clients
* Provide a central Redfish/Swordfish database for abstractly associating edges (fabrics links) to nodes (clients and resources)
* One set of abstract tools for all compute fabrics
  + In current HPC and Cloud system management, controlling compute fabrics requires a different set of options and control knobs depending upon the type of fabric used
  + Future HPC and Cloud system management will benefit from the ability to aggregate and subdivide fabrics regardless of the fabric types

# Architecture of the OpenFabrics Management Framework



* Clients and Client Applications are placed on the left side of the diagram
  + Clients in the administration domains are considered to be HPC and Cloud system Users, their batch and interactive jobs, System Administrators, and other machine stakeholders
    - ‘Bare metal’ machines
    - Virtual machines and Containers
  + Client in the application domain consist of Batch and Interactive jobs, libraries, monitoring systems, compilers, user environments
* The OpenFabrics Management Framework provides RESTful services, in the middle of the diagram
  + Resource inventory consists of available memory, available processors, available switches, available accelerators, and fabric connections to consolidate and make connections
    - CXL-3.0 compatible resources
      * PCIe-6.x with a peer-to-peer CPU bus addendum
      * Flash memory can be subdivided and associated with CPUs and accelerator cards
      * Peer-to-Peer switching allows for any type of switch topology from Fat-Trees to Rings to Dragon Fly topologies
    - Aggregation of resources under groups
      * A set of resources can be grouped under a set of routes and switches
      * The OFMF will know how to reach each resource and make abstract connections as a client makes resource requests
    - Fabric connections are made abstractly
      * Each fabric connection uses the same set of tools
  + Redfish Tree Management
    - One set of RestFul APIs provides information on connections, clients, and resources
      * Information is provided in the same manner for each resource and connection
    - Aggregation and disaggregation of resources are provided with the same set of tools
  + Fabric and resources configuration
  + Authentication and Access control
    - User authentication
    - Multi-tenancy
  + Events and Logs
    - Flapping connections
    - Adding and removing fabrics
    - Dynamically adding and subtracting fabric connections
  + Systems composability
* Right side of diagram consists of hardware
  + Fabrics and Fabric Managers
    - Ethernet, InfiniBand, Slingshot, OmniPath, CXL-3.0, Gen-Z, etc.
    - Information and control is provided by a Fabric Manager
  + CXL-3.0 compatible hardware
    - CPUs
    - Fabric Attached Memory
    - Accelerator cards

# OFM Fabric Agents

* Agents provide an interface from the OpenFabrics Management Framework to the physical hardware
  + Translation is provided from Redfish and Swordfish to physical hardware commands
  + Information on fabrics and resources is translated back to the OpenFabrics Management Framework in real-time
* Agent registration to OFM
  + Define how an agent authenticates with OFM
    - Does it use the regular redfish authentication?
  + Define how the OFM RF tree is populated when a new agent comes up or as a response to an agent event
    - Example: Agent registers using an Event on the Fabrics object and the OFM scans the agent RF tree to populate its own
    - Alternative is the Agent performs a number of POST calls against the

# Resources Configuration

* One set of tools to manage Fabric Attached Memory, accelerator cards, CPUs, High-Bandwidth memory

# Fabric Configuration

* One common set of tools to provide abstract management of fabric resources
* Fabric links and their capabilities are displayed and managed as disaggregated resources
* How-to connect resources in the fabric

# Authentication and access control

* Authentication
  + Redfish/Swordfish provides APIs to provide security for resources and fabric connections
* Access Control
  + Multi-tenancy
  + Shared Fabric-Attached-Memory resources
  + NVME over Fabric access control

# Systems composability

* Composability Layer provides clients with an intelligent interface to the OpenFabrics Management Framework
  + Real-time fabric resource monitoring
  + Composition policies
  + Resource control
    - Attaching resources and fabrics to clients
* Define how the OFM supports composability
  + Keeping ResourceBlock objects in the RF tree
  + Bookkeeping at the resource block level
* Define how agents specify QoS related or fabric specific details in existing RedFish objects
  + e.g., use the oem field in each object or propone changes to redfish

# Events and logs

* Configuration and forwarding of events
  + Fabric links added and subtracted
  + Resources added and subtracted
  + Access granted and removed to clients
* Define a set of mandatory events needed for supporting registration of clients/agents
* Required events subscription/generation for Clients and Agents

# Security

* Support for confidential computing

# Use cases

* Show a few OFM usage cases with and without composability enabled
  + CXL2.0/3.0
  + GenZ?
  + Do we have something in mind for IB?