Looking for a Swiss knife for storage ecosystem management?

A comparative study of SMI-S, Redfish and Swordfish

Anand Nagarajan and Sona Nagarajan
Microsemi
Agenda

- Storage management standards
- Modelling ideology
- Modelling architecture
- Merits and demerits
- Standardization and customization
Why Standardize?

- Multiple components by different vendors in a server
- Different management interfaces
- Different data exchange formats
- Rapid scalability
- Security is always a concern
Standardization

- Standard way of modelling all components of an IT environment
- Standard data format for representing attributes and events
- Standard interface for performing any operation
- Ease of programming, scripting, and human readability
Storage Management Standards

- CIM
- Swordfish
- Redfish
The Storage Management Initiative Specification (SMI-S) from SNIA standardizes and streamlines storage management functions and features into a common set of tools.

Operations include identifying and modifying attributes of components, discovery, security, virtualization, performance, and fault reporting.

Provides manageability of both hardware and software components.

Extends the Common Information Model (CIM) of DMTF with the data format as CIM-XML.

Uses MOF syntax to describe classes.
SMI-S: Modelling Ideology

- Modelled on CIM, which is an object-oriented model
- Managed elements are represented as CIM classes that include properties and methods to represent management data and functions
- The CIM schema is an implementation of CIM to model various classes needed in an IT environment such as computer systems, networking, applications, storage, and more
- Users can extend the CIM schema by inheriting and extending existing classes to better describe their component
- EnumerateInstance, ModifyInstance, InvokeMethod and DeleteInstance on objects of CIM classes.
SMI-S: Modelling Architecture

- CIM_ManagedElement
  - CIM_ManagedSystemElement
    - CIM_LogicalElement
      - CIM_StoragePool
        - OEM_StoragePool
SMI-S: Advantages and Disadvantages

- **Advantages**
  - Evolved and stable standard
  - Preferred standard in virtualization
  - Independence from platform, programming language, and compiler
  - Clients available in all frequently used programming languages
  - Products from multiple vendors can be treated in the same way
  - Reliability, security, and better quality of product

- **Disadvantages**
  - In-band only management solution
  - Numerous levels of inheritance and association classes
  - Data parsing and conversion needed because of XML data format
  - Clients need to have in-depth knowledge of SMI-S
Redfish

- DMTF’s Redfish is an open standard specification and schema for simple and secure management of modern scalable platform hardware.
- Specifies a RESTful interface, uses HTTPS as the transport protocol, and utilizes JSON and OData as the data format.
- Provides ways to manage resources, discovery, authentication, authorization, event reporting, and task handling.
Redfish: Modelling Ideology

- Everything is a self-contained resource with no inheritance or polymorphism
- Resources are linked from the service entry point /redfish/v1
- Major homogeneous resource types are structured together to represent collections
- Resources are broadly classified under one of the three views:
  - Physical view—resources grouped under Chassis
  - Logical or data view—resources grouped under Systems
  - Management view—resources grouped under Manager
- Reduce network traffic
Redfish: Modelling Architecture

- /redfish/v1
- /redfish/v1/Systems
- /redfish/v1/Chassis
- /redfish/v1/Managers
- /redfish/v1/SessionService
- /redfish/v1/AccountService
- /redfish/v1/TaskService
- /redfish/v1/EventService
- Storage
- Volumes
- Drives
- Enclosures
Redfish: Advantages and Disadvantages

- **Advantages**
  - Out-of-band management through MCTP
  - In-band management through device driver
  - JSON OData format and flat resource schema
  - Applications will be abstracted from communication path as long there is an HTTPs server
  - Client-oriented data representation
  - Well-defined schema, including OEM extensions, foster extensibility

- **Disadvantages**
  - Storage management modelling needs better handling
  - Newer standard that requires better adoption and wider contribution
Swordfish

- The SNIA Swordfish specification helps to provide a unified approach for the management of storage and servers in hyperscale and cloud infrastructure environments.
- Extends DMTF Redfish specification and leverages all the services and data formats specified by Redfish.
- Seamlessly manages storage equipment and storage services in addition to server.
- Provides functionality that simplifies the way storage is allocated, monitored, and managed.
Swordfish: Modelling Ideology

- Leverages and extends DMTF Redfish specification
- Refactors and leverages SMI-S schema into a simplified client-oriented model
- Provides class of service-based provisioning and monitoring
- Covers block, file, and object storage
- Extends traditional storage to include converged environment
Swordfish: Modelling Architecture

/redfish/v1

/redfish/v1/Systems

/redfish/v1/Chassis

/redfish/v1/AccountService

/redfish/v1/TaskService

/redfish/v1/SessionService

/redfish/v1/StorageSystems

/redfish/v1/SessionService

/redfish/v1/Chassis

/redfish/v1/Managers

/redfish/v1/AccountService

/redfish/v1/TaskService

/redfish/v1/EventService

/redfish/v1/StorageServices

/StorageGroups

/StoragePools

/Storage

/Drives

/Volumes

/Endpoints

/Enclosures
Swordfish: Advantages and Disadvantages

- **Advantages**
  - All the advantages of Redfish are applicable
  - Class of service-based storage/resource provisioning and monitoring
  - Converged IT environment and traditional storage domain support
  - Power of SMI-S in a simplified client-friendly format
  - Scalable solution

- **Disadvantages**
  - Newer standard that requires better adoption and wider contribution
What Have We Done?

- Redfish and Swordfish based client and server framework to manage different families of Microsemi storage controllers in a direct attach storage configuration
- Integration of Redfish client-server framework with OpenStack Horizon and our storage management GUI
- Redfish-based server plugins to seamlessly manage different Microsemi products
- SMI-S provider for managing Microsemi storage controllers
- Integration of SMI-S provider with our management GUI and vSphere web client plugin
Customize applications to suit diverse end-user needs
- Thinner clients with better user design
- Standardized common communication path, model, and data format
- Customize component data by extending standard schema
Migrating Existing Systems to Newer Standards

Identify
- Components
- Attributes
- Operations
- Interactions
- Persistence

Extend
- Components

Choose
- Standard
- WebServer
- CIMOM

Include/Create
- Schema
- Composition
- Associations

Implement
- Software suite
Food for Thought

- Automated error notification, handling, and correction
- User-focused interface providing most relevant information
- Convergence of diverse component management of a system
- API driven design
- Self-learning component management
References

- https://www.dmtf.org/standards/redfish
- http://www.snia.org/swordfish
- https://www.snia.org/forums/smi/tech_programs/smis_home
- http://www.dmtf.org/standards/cim
Contact Us

- Anand Nagarajan—anand.nagarajan@microsemi.com
- Sona Nagarajan—sona.nagarajan@microsemi.com
- Microsemi website—www.microsemi.com