A Deep Dive Sea Trek With SNIA Swordfish™

Richelle Ahlvers
Principal Storage Management Architect, Broadcom Inc.

SNIA Scalable Storage Management (SSM) Technical Work Group Chair
Disclaimer

- The information in this presentation represents a snapshot of work in progress within SNIA
- This information is subject to change without notice.
- For additional information, see the SNIA website: www.snia.org/swordfish
Abstract

- Swordfish™ is an extension of the DMTF Redfish® specification developed by the Storage Networking Industry Association (SNIA) to provide a unified approach for the management of storage equipment and services in converged, hyper-converged, hyperscale and cloud infrastructure environments, making it easier for IT administrators and DevOps to integrate scalable solutions into their data centers.

- This session builds on the intro sessions to Redfish and Swordfish, presenting more detail about the Swordfish Class of Service concepts, structure and usage;
- A look at Swordfish schema (both CSDL and JSON),
- Usage of the two Swordfish Service Configurations,
- A preview of the emerging Swordfish profile definition work.
Swordfish Concepts
Primary Swordfish Elements

- **ClassOfService**
  - A choice of utility or warranty offered to customers by a service. Defined by selecting from available LinesOfService.

- **StorageService**:
  - Represents a service that provides ClassOfService based provisioning, management, and monitoring for logical storage and associated resources.

- **StoragePools**
  - Storage capacity that can be used to produce volumes or other storage pools with a specified class of service.

- **Volume**
  - Block addressable storage that is conformant to a ClassOfService.

- **StorageGroup**
  - A set of volumes that are managed as a group; e.g., with the same consistency requirements.

- **Filesystem**
  - File-addressable storage that is conformant to a ClassOfService.

- **Fileshare**
  - A shared set of files with a common directory structure that is exported for use by remote systems.
Storage Services

- Contains a list of Storage Service instances that manage the different storage related functionality
- Focus of StorageService is on the logical aspects of the storage (such as classes of service, volumes, file systems / file shares, storage pools)
Class of Service
Why Class of Service

- Modern volumes, file systems, and object stores are often built on complex, multiply layered algorithms and implementations
  - The internals of storage servers are quite complex and vary by product
- Configuration of these elements, can be complex and may require detailed implementation and implementations can change over time with unexpected consequences
- Class of Service addresses these concerns.
- An implementation advertises a set of supported configuration options that will deliver desired levels of service in terms that are meaningful to the user.
- Provisioning a resource to a Class of Service allows each user to get expected levels of service while enabling each implementation to set the configuration correctly to deliver the requested levels of service.
Class of Service

- Swordfish introduces provisioning based on a requested class of service
  - A Class of Service specifies one or more desired levels of service
- Implementation independence
  - Ex 1: Same level of performance might be delivered using a single drive with faster drive technology or via striping multiple drives.
  - Ex 2: Thin provisioning via various vendor specific techniques
  - Ex 3: The best connectivity choices for throughput and bandwidth vary with implementation
  - Ex 4: Where and how to best protect storage
- Class of Service based systems are expected to monitor compliance and to creates notifications when out of conformance
Defining Classes of Service

- Classes of Service are composed of Lines of Service
- Types: Data Protection, Data Security, Data Storage, Connectivity, Performance
- Properties of each are constrained by …LoSCapabilities

ClassesOfService

...<id>

1..n

Collections of LinesOfService

...LoSCapabilities

LineOfService Capability
Supported capabilities of the specific system

DataProtectionLineOfService 0..n
DataSecurityLineOfService 0..n
DataStorageLineOfService 0..n
IOConnectivityLineOfService 0..n
IOPerformanceLineOfService 0..n
Capability Types (XXXLoS Capabilities)

- Data Protection – replicas for recovery
  - Type of replica, schedule, geographic requirements
- Data Security – encryption, authentication, anti-virus, sanitization
- Data Storage – high availability, R/W access, thin provisioning
  - Recovery time requirements, thick/thin provisioning
- IO Connectivity – storage interface characteristics
  - Connection type, media class
- IO Performance – workload description
  - Guaranteed workload / latency, workload throttling
Simple Example:

- Use a single capability property to create two possible classes of service
  - IOConnectivityLoSCapabilities. SupportedAccessProtocols (e.g., drive type)
- Create LinesOfService that includes this property
- Create two classes of service: One for NVMe drives, one for SAS / SATA drives
Example: Two Classes Of Service

```
"@odata.id": "/redfish/v1/StorageServices/1/ClassesOfService/HighCapacity",

"@odata.type": 
"#ClassOfService_1_0_0.ClassOfService",

"Name": "HighCapacity",

"Description": "HighCapacity Class Of Service (SATA)",

"Id": "HighCapacity",

"ClassOfServiceVersion": "1.01.00",

"Status": { "State": "Enabled", "Health": "OK" },

"LinesOfService": {

"IOConnectivityLineOfService": {

"Name": "Serial Attached ATA",

"AccessProtocol": "SATA"

}

}

"@odata.context": "/redfish/v1/$metadata#StorageServices/1/ClassesOfService/HighPerformance",

"@odata.id": "/redfish/v1/StorageServices/1/ClassesOfService/HighPerformance",

"@odata.type": 
"#ClassOfService_1_0_0.ClassOfService",

"Name": "HighPerformance",

"Description": "HighPerformance Class Of Service (NVMe SSDs)",

"ClassOfServiceVersion": "1.01.00",

"Status": { "State": "Enabled", "Health": "OK" },

"LinesOfService": {

"IOConnectivityLineOfService": {

"Name": "NVMe",

"AccessProtocol": "NVMe"

}

}
```

Using Classes Of Service

- Once ClassesOfService exist (either pre-existing from vendor or constructed from capabilities -> LinesOfService -> ClassesOfService):
  - Create StoragePools using specified Classes of Service
  - When allocating volumes, use the StoragePools’ Class of Service attributes to determine which to use to request capacity
Example: Create Volume

`POST /redfish/v1/StorageServices(1)/Volumes/`

```json
{
    "Name": "Volume56",
    "CapacityBytes": 1099511627776,
    "Links": {
        "ClassOfService": {
            "odata.id": "/redfish/v1/StorageServices(1)/ClassesofService(HighPerformance)"
        }
    }
}
```

Link available at www.snia.org/swordfish
Schema Primer
Schema File Types

- Currently two types of schema files exist: CSDL and JSON
  - CSDL (one file with all versions)
    - Common Schema Definition Language
    - XML formatted
    - Standardized by OASIS to support the OData standard
  - JSON (one file per version)
    - Generated by scripts from the CSDL schemas
- Both sets of schemas are functionally equivalent
- Using one type over the other is a matter of preference on the part of the developer

**NOTE:** Both Redfish and Swordfish are adding OpenAPI support
Schema File Rules

- Each resource (entity type) is described by a single schema file (in CSDL)
  - Each schema file may pull in external definitions
  - In JSON, a version is created for each file version (including unversioned)
- Schema files also contain additional supporting structures (e.g., ComplexTypes, Enums)
  - Shared structures are referenced by the namespace they are defined in: An Enum in the Resource_v1.xml file would be referenced as Resource.Enum
- Redfish and OData Annotations define and constrain schema elements
- Schema contain a set of OData properties to provide additional identifying information
  - Each response contains an "@odata.type" property to provide top level decoding information
    - "@odata.type" is broken down as "#Namespace.Entity"
    - Example: "#Volume.v1_2_0. Volume" means it’s a “Volume” entity in the “Volume.v1_2_0” namespace
Using Schema Files

- Generic clients may dynamically parse schema files as they are interacting with a service in order to automatically build data models.
- Purpose built clients may not necessarily read schema files directly.
  - Developers may reference schema files when writing these clients to understand what resources and properties are available.
- All standard schema files are (re-)published on the DMTF website.
  - DMTF schema published directly
  - SNIA schema “republished”
  - OEM schema may also be (and are recommended by SNIA Scalable Storage Management TWG to be) “republished” on the DMTF site.
- Clients and services may have local copies.
CSDL Schema Example

<edmx:Edmx xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx" Version="4.0">

  <edmx:Include Namespace="Org.OData.Core.V1" Alias="OData"/>
</edmx:Reference>

  <edmx:Include Namespace="Org.OData.Capabilities.V1" Alias="Capabilities"/>
</edmx:Reference>

<edmx:Reference Uri="http://redfish.dmtf.org/schemas/v1/Resource_v1.xml">
  <edmx:Include Namespace="Resource.v1_0_0"/>
</edmx:Reference>

<edmx:Reference Uri="http://redfish.dmtf.org/schemas/v1/RedfishExtensions_v1.xml">
  <edmx:Include Namespace="RedfishExtensions.v1_0_0" Alias="Redfish"/>
</edmx:Reference>

<edmx:DataServices>

  <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="Session">
    <EntityType Name="Session" BaseType="Resource.v1_0_0.Resource" Abstract="true">
      <Annotation Term="OData.Description" String="..."/>
      <Annotation Term="OData.LongDescription" String="..."/>
    </EntityType>
  </Schema>

</edmx:DataServices>
CSDL Schema Example (cont.)

```xml
<Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="Session.v1_0_0">
  <Annotation Term="Redfish.OwningEntity" String="DMTF"/>
  <Annotation Term="Capabilities.InsertRestrictions">
    <Record>
      <PropertyValue Property="Insertable" Bool="false"/>
    </Record>
  </Annotation>
  <EntityType Name="Session" BaseType="Session.Session">
    <Property Name="UserName" Type="Edm.String">
      <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
      <Annotation Term="Redfish.RequiredOnCreate"/>
      <Annotation Term="OData.Description" String="..."/>
      <Annotation Term="OData.LongDescription" String="..."/>
    </Property>
    <NProperty Name="Password" Type="Edm.String">
      <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
      <Annotation Term="Redfish.RequiredOnCreate"/>
      <Annotation Term="OData.Description" String="..."/>
      <Annotation Term="OData.LongDescription" String="..."/>
    </Property>
  </EntityType>
</Schema>
```
CSDL Schema Example (cont.)

```xml
<Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="Session.v1_1_0">
  <EntitySet Name="Session" BaseType="Session.v1_0_0.Session"/>

  <Property Name="Actions" Type="Session.v1_1_0.Actions" Nullable="false">
    <Annotation Term="OData.Description" String="..."/>
    <Annotation Term="OData.LongDescription" String="..."/>
  </Property>
</EntityType>

<ComplexType Name="Actions">
  <Annotation Term="OData.AdditionalProperties" Bool="false"/>
  <Annotation Term="OData.Description" String="..."/>
  <Annotation Term="OData.LongDescription" String="..."/>
  <Property Name="Oem" Type="Session.v1_1_0.OemActions" Nullable="false"/>
</ComplexType>

<ComplexType Name="OemActions">
  <Annotation Term="OData.AdditionalProperties" Bool="true"/>
  <Annotation Term="OData.Description" String="..."/>
  <Annotation Term="OData.LongDescription" String="..."/>
</ComplexType>

</Schema>
</edmx:DataServices>
</edmx:Edmx>
```
JSON Schema Example

```json
{
  "$schema": "http://redfish.dmtf.org/schemas/v1/redfish-schema.v1_2_0.json",
  "title": "#Session.v1_1_0.Session",
  "$ref": "#/definitions/Session",
  "definitions": {
    "Session": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "@odata.context": { "$ref": "http://.../odata.4.0.0.json#/definitions/context" },
        "@odata.id": { "$ref": "http://.../odata.4.0.0.json#/definitions/id" },
        "@odata.type": { "$ref": "http://.../odata.4.0.0.json#/definitions/type" },
        "Oem": {
          "$ref": "http://.../Resource.json#/definitions/Oem",
          "description": "...",
          "longDescription": "..."
        },
        "Id": {
          "$ref": "http://.../Resource.json#/definitions/Id",
          "readOnly": true
        }
      }
    }
  }
}
```
"Description": {
  "anyOf": [
    {
      "$ref": "http://.../Resource.json#/definitions/Description",
      "type": "null"
    },
    "readonly": true
  ],
  "readonly": true
},
"Name": {
  "$ref": "http://.../Resource.json#/definitions/Name",
  "readonly": true
},
"UserName": {
  "type": [ "string", "null" ],
  "readonly": true,
  "description": "...",
  "longDescription": "...
},
"Password": {
  "type": [ "string", "null" ],
  "readonly": true,
  "description": "...",
  "longDescription": "...
}
JSON Schema Example (cont.)

"Actions": {
    "type": "object",
    "additionalProperties": false,
    "properties": {
        "Oem": {
            "type": "object",
            "additionalProperties": true,
            "properties": {},
            "description": "...",
            "longDescription": "..."
        }
    },
    "description": "...",
    "longDescription": "..."
}

"required": [ "Id", "Name" ],
"requiredOnCreate": [ "UserName", "Password" ],
"description": "...",
"longDescription": "..."
Deploying Swordfish: ISC and HSC Configurations
Storage Deployment Options

- Direct-attach progression:
  - Server with direct connect drives
    - Use SimpleStorage
  - Server with either HBA with connected drives or RAID card with small # of drives
    - Use Storage+Volume+Drives
  - Increase scale (# of drives) or functionality
    - Use Storage + StorageServices == Swordfish (ISC)

- Standalone Configurations:
  - External storage, SDS, Layered or composed systems
  - Use HSC (Swordfish)
Starting with Redfish: Basic Compute

HTTP GET /redfish/v1/Systems/CS_1/Processors/2

Collection of Systems
"Logical view"

Collection of Chassis
"Physical view"

Collection of Managers
"Mgmt hierarchy"

Managed By

BMC

LogService

NW protocol

Power

Thermal

Platform HW Mgmt

Computer Systems

Computer System

Processes

Memory

SimpleStorage

EthernetInterfaces

Compute

Schemas

Sessions

Accounts

Events

Registries

Tasks
The Storage/StorageController in ComputerSystem hosts the StorageServices. It is capable of creating StorageServices and additional Swordfish functionality.
Hosted Service Configuration

Service Root:
- /redfish/v1
  - Root
- Schemas
- Sessions
- Accounts
- Events
- Registries
- Tasks

Collection Resource:
- /redfish/v1/StorageServices
  - Collection of StorageServices
  - "Logical view"
- /redfish/v1/StorageSystems
  - Collection of StorageSystems
- /redfish/v1/Systems
  - Collection of Systems
  - "Logical view"
- /redfish/v1/Chassis
  - Collection of Chassis
  - "Physical view"
- /redfish/v1/Managers
  - Collection of Managers
  - "Mgmt hierarchy"

Single Resource:
- /StorageService/<id>
  - Storage Service
- /Systems/<id>
  - Computer System
- /Chassis/<id>
  - Chassis
- /Managers/<id>
  - BMC (or SW Manager)
  - Managed By

A StorageSystem *is* an instance of a ComputerSystem (with a small set of additional properties defined).
Features and Profiles
How Do Clients Determine Which Swordfish Implementations They Can Support?

- Runtime: clients can look for advertised implemented features
- During development, look at detailed profiles that map to these features

- Features and profiles used for:
  - “Advertising” implemented functionality
  - Clients use to determine what features to require for different configurations
  - Certification / Conformance requirements, testing
    - Swordfish CTP Certification
    - (Future) OCP, ODCC Storage
    - (Planned) EnergyStar Requirements: Orthogonal to functionality profiles
      - Energy and power metrics
      - Controls for on-demand instrumentation
Advertised Features

- For supported service types…
  - Services == block, file, object
- Features define coarse-grained sets of required functionality:
  - Standard Features:
    - Discovery / Inventory
    - Events
    - Performance Instrumentations
    - Basic Provisioning
  - Advanced Features:
    - Local replication provisioning
    - Remote replication provisioning
    - Advanced Configuration Management
    - Mapping and Masking for Provisioning
Profiles

- Profiles are detailed descriptions for storage service (provider) implementers, inputs to tests, etc., that describe down to the individual property level what functionality is required in order to advertise features.
- There will be multiple profile definitions, for different types of storage configurations, that map to the same feature: Block.Provisioning, File.Provisioning.
- Profiles are predominantly a development tool.
Swordfish Specs and Technical Content... In 2018

- v1.0.6 Technical Position Release in (WIP in Feb 2018, TP in May 2018)
  - Introduction of two StorageSystem models:
    - Hosted Service Configuration and Integrated Service Configuration
  - Schema updates, Spec section additions, User’s guide updates: new use cases for on-demand replicas
- Q4 2018 Releases:
  - Updated Swordfish mockups: swordfishmockups.com
  - v1.0.7 Swordfish WIP Release:
    - Enhanced Class of Service Capabilities for Spare Capacity Management, Rebuild Management, Volume types
  - White Paper for Spare Management
  - WIP Profile Development: Basic Swordfish Support
- Future Functionality
  - Storage-specific security roles
  - Enhanced profiles for SNIA Alliance partner organizations
  - Functionality alignment across DMTF, NVMEExpress/NVMe-MI and SNIA
  - Object Storage
Resources
Swordfish Info: www.snia.org/swordfish

- Resources
  - Specifications
  - User’s Guide
  - GitHub for Swordfish Tools
  - Practical Guide
  - Other Documentation
- Swordfish Mockups Site
  - ISC and HSC configurations
  - Block vs file configurations
  - Small and large configurations
- Education/Community
  - Whitepapers, Presentations
  - YouTube shorts & Webinars
- Participate
  - Join SNIA and the SSM TWG
  - Implement
Open Source Tools and Infrastructure Development

- Available: [http://github.com/snia](http://github.com/snia)
  - Swordfish Emulator Extensions
    - Extends the Redfish emulator – adds all Swordfish schema (behave like dynamic objects)
  - Basic Swordfish Web Client
    - Discover, display and edit Swordfish services
  - DataDog and Power BMI Client Sample Dashboards
    - Sample implementations show integration concepts with sample code:
      - PowerBI: Point-in-time dashboard; Datadog: Data trending dashboard
Documentation and Supporting Materials

- Online Practical Guide
  - SNIA Swordfish Practical Guide
- Swordfish School:
  - Swordfish School Playlist (YouTube)
- Swordfish API Specification
- Webcasts
How to Participate: Shaping the Standard

- Find pointers to the latest technical content:
  - [http://snia.org/swordfish](http://snia.org/swordfish)
  - [http://www.snia.org/publicreview#swordfish](http://www.snia.org/publicreview#swordfish)
- Join the SSM TWG
  - By joining the SNIA and SSM TWG, you can shape the standard: [https://members.snia.org/apps/org/workgroup/ssmtwg](https://members.snia.org/apps/org/workgroup/ssmtwg)
- Through the SNIA feedback portal, providing feedback on “Work In Progress”
  - As the group produces “Works In Progress”, you can provide feedback at [http://www.snia.org/feedback](http://www.snia.org/feedback)
Q&A