



Introduction to SNIA Swordfish™ — Scalable Storage Management

Richelle Ahlvers

Principal Storage Management Architect Broadcom Inc.

SNIA Scalable Storage Management (SSM) Technical Work Group Chair

SNIA at-a-glance



160
unique member
companies



3,500
active contributing
members



50,000
IT end users & storage
pros worldwide

Learn more: snia.org/technical



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 presentation shows how Swordfish extends Redfish and provides an overview of basic Swordfish concepts.



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



There are 3 hard problems...

- There are 3 hard problems in tech today

0. Cache coherence
1. Naming things
1. Off by one errors



There are 3 hard problems...

- There are 3 hard problems in tech today

0. Cache coherence
1. Naming things
1. Off by one errors
7. Async callbacks

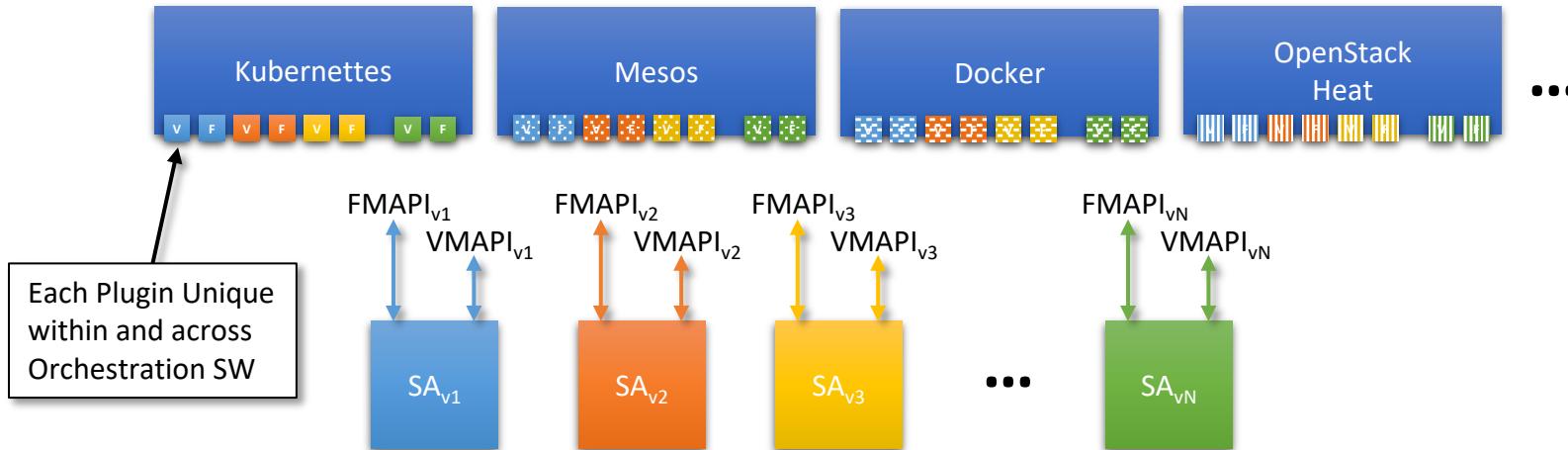


What are the Drivers for SNIA Swordfish™?

- Customers (and vendors) have asked for improvements in storage management APIs
 - Make them simpler to implement and consume
 - Improve access efficiency
 - Fewer transactions, with more useful information in each
 - Provide useful access via a standard browser
 - Expand coverage to include converged, hyper-converged, and hyper-scale
 - Provide compatibility with standard DevOps environments



Problems – Orchestration Interaction



Key
SA = Storage Array
SAv1 = Storage Array Vendor 1
VMAPIv1 = Volume Management API Vendor 1
FMAPIv1 = File Management API Vendor 1

- Orchestration components need plugins for every FMAPI and VMAPI
- Orchestrations layers do not have easy API for plugins
- Currently plugin support is built into the orchestration layer itself
 - Orchestration layers change frequently requiring constant
 - Every orchestration layer will require unique plugins



More Problems – Storage Array Vendors

- To write the volume and file plugins Storage Vendors have to:
 - ◆ Possibly write more than one set per array type per orchestration, as different orchestration versions have evolved their storage support requiring different plugins for different versions
 - ◆ Develop strong orchestration product expertise as the plugins are very orchestration specific
 - ◆ These plugins will have to run on virtually every node in the cluster
 - ◆ Because these plugins may be tied into the orchestration software they may end up in tree for the orchestration SW



More Problems

- Some orchestration layers may leave array selection as an app/container decision
- No common storage scheduling
 - ◆ DevOps should not pick data locations directly
 - ◆ No standard based definition of classes of service
- No automated discovery

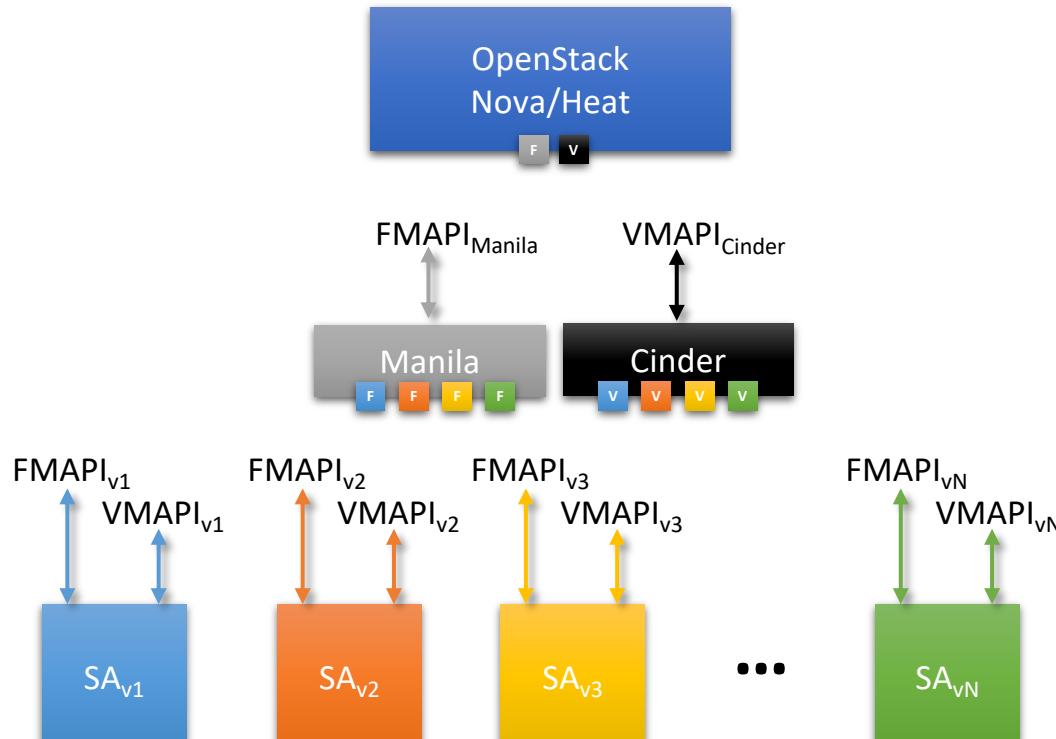


OpenStack Mostly Solved This

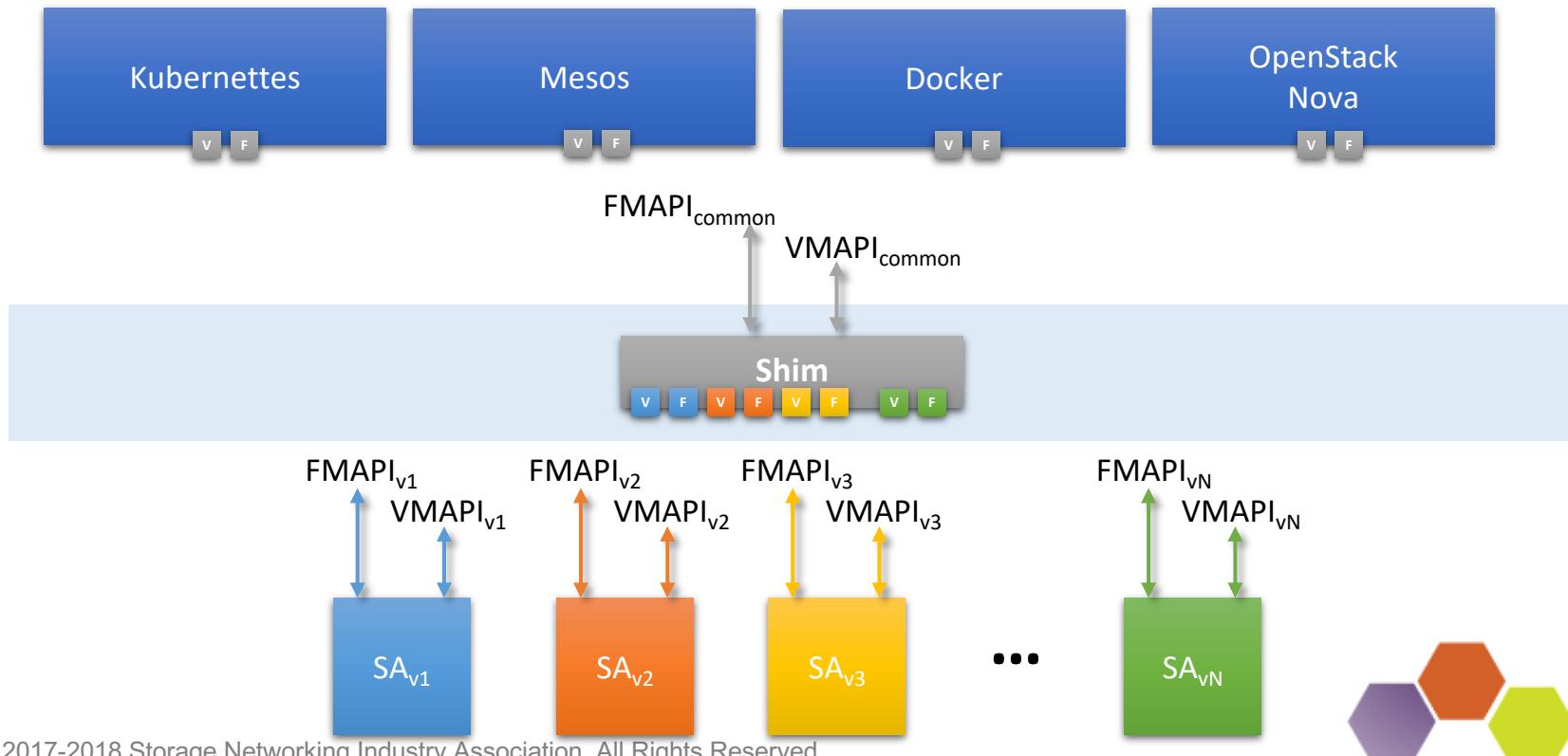


- Created a shim layer:
 - Cinder API – Volume Management API
 - Manila API– File Management API
- Each Vendor required to created Cinder and Manila Plugin
 - Cinder/Manila are specific to Openstack
 - Change infrequently
 - Can not be used easily outside of Openstack
 - > Being retro fitted to a stand alone mode for other orchestrations
- Cinder has a filter scheduler
 - Based on characteristics and weights will select best array for the app request
- Does not solve all the issues
- Endorsed and deployed widely by the community

The Openstack Solution



A Common Shim solution



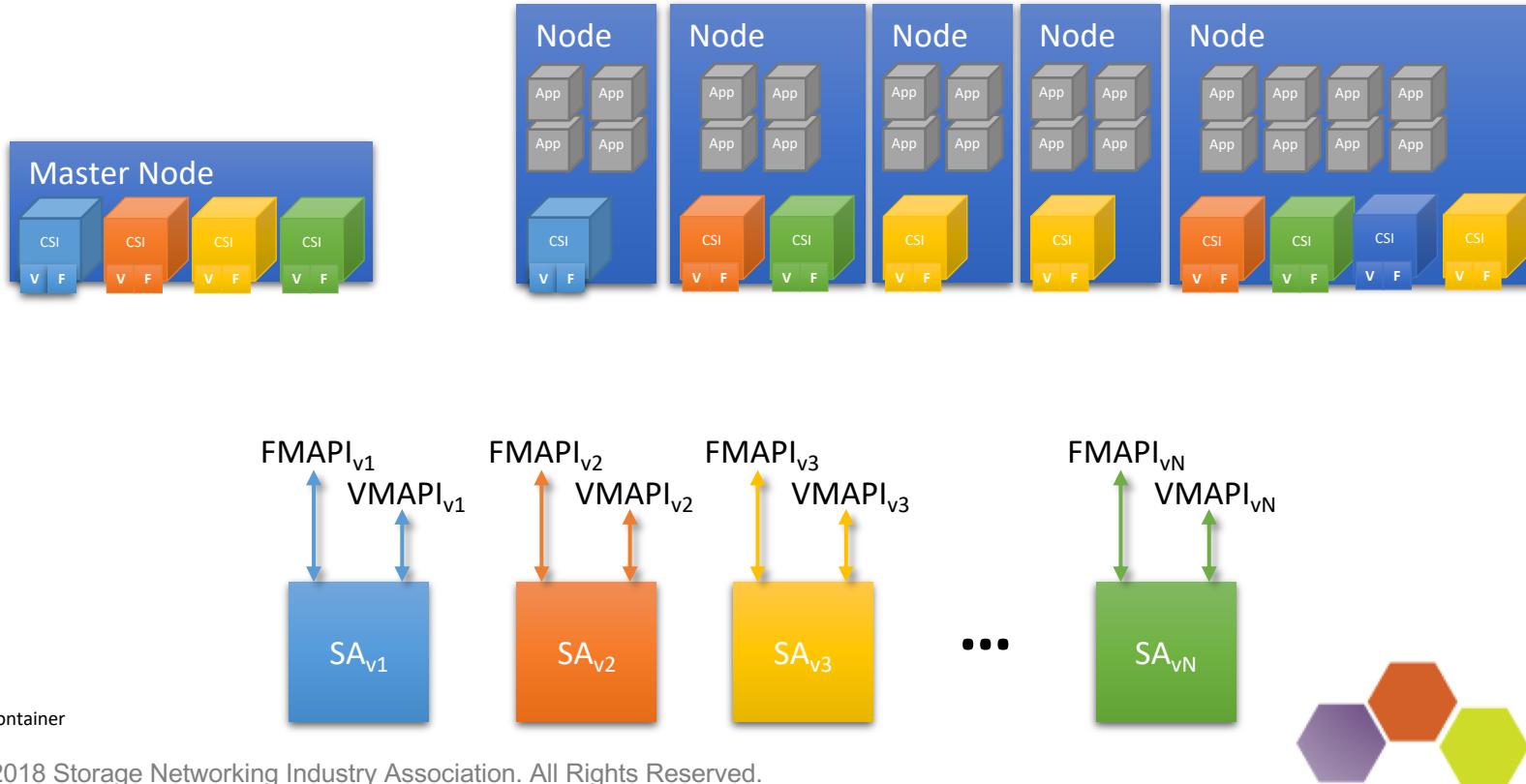
Container Storage Interface

➤ Orchestrators last year determined this was a problem

- ◆ Set out to define a simple shim service has
 - > Common northbound
 - > Stable southbound for array plugins
- ◆ Cooperative effort – One API to rule them all
 - > Kubernettes
 - > Docker
 - > Mesos



CSI Direction



CSI Problems

- CSI defines an abstraction that creates an opaque conduit to Storage Array
 - ◆ Orchestration layer is only concerned with size and name at this point
 - ◆ Container writer must provide array specific blob
- Simple provisioning interface only
 - ◆ Even snapshots and replications deferred
 - ◆ No object support
 - ◆ No tenancy
 - ◆ No storage scheduling
 - > No classes of service
 - ◆ No unified management APIs
 - > No events management
 - > No log management
 - > No performance monitoring
 - > No migration services
- If there were only a unified model that dealt with all these
 - ◆ Perhaps a tail-walking, bill-shaking, powerful beast



The SNIA Swordfish™ Approach



■ The What:

- Refactor and leverage SMI-S schema into a simplified model that is client oriented
- Move to Class of Service based provisioning and monitoring
- Cover block, file and object storage
- Extend traditional storage domain coverage to include converged environments (covering servers, storage and fabric together)

■ The How:

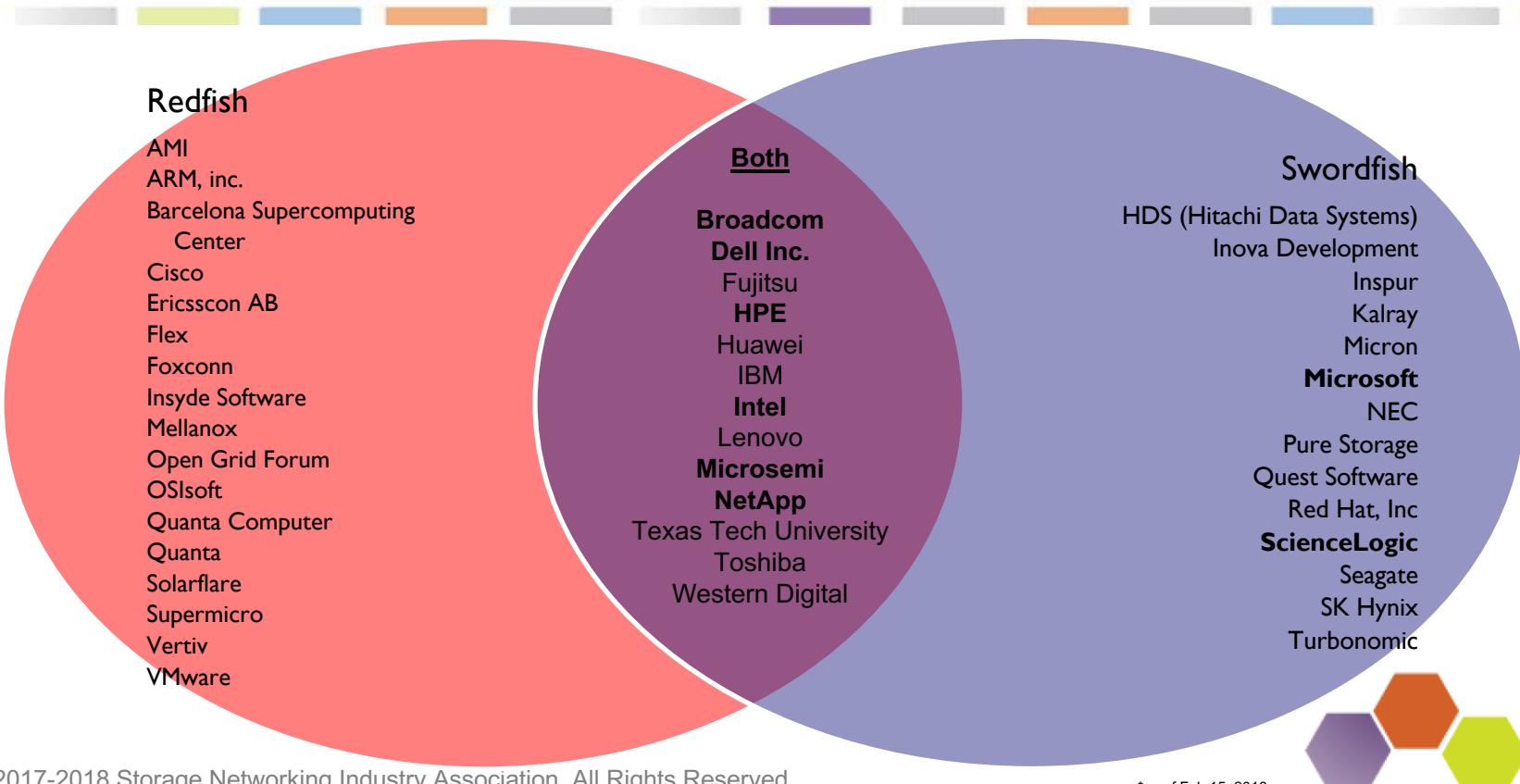
- Leverage and extend DMTF Redfish Specification
- Build using DMTF's Redfish technologies
 - RESTful interface over HTTPS in JSON format based on OData v4
- Implement Swordfish as an ***extension*** of the Redfish API



Who is Developing Redfish and Swordfish*?



STORAGE
MANAGEMENT



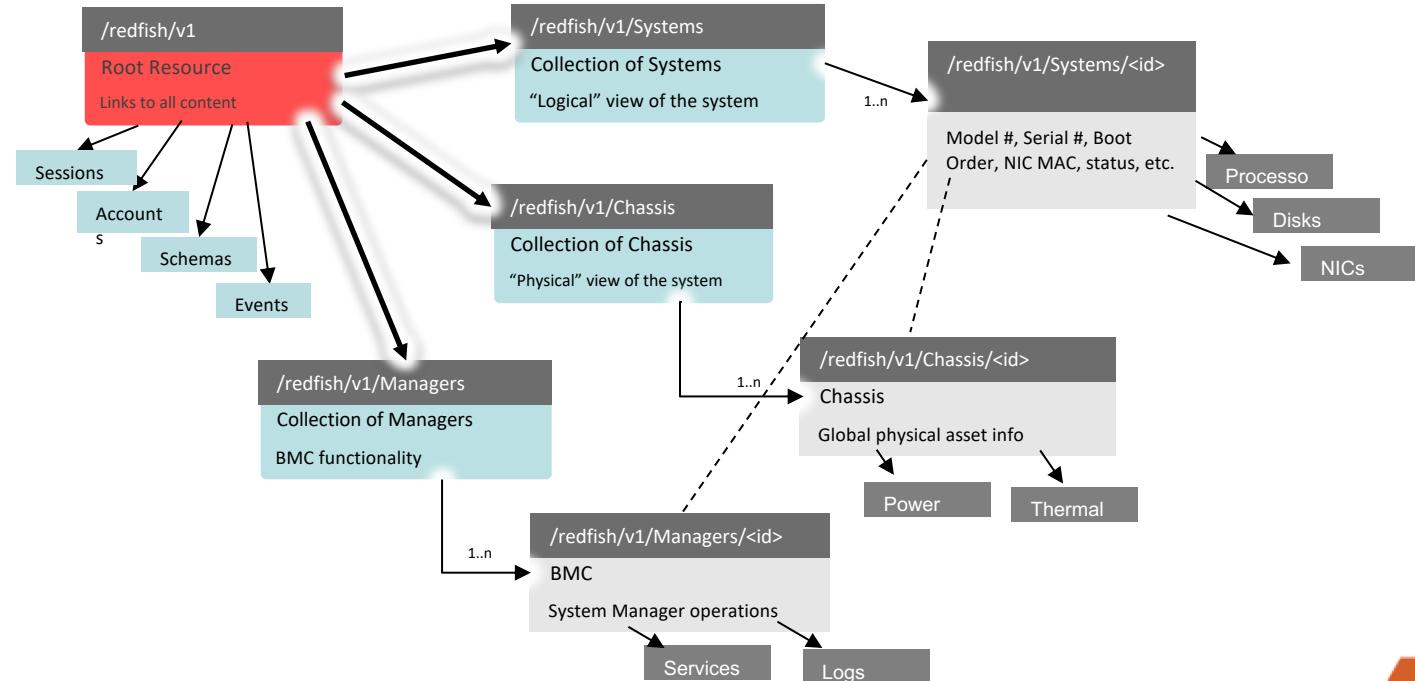
Functionality Included in the Swordfish v1.0.x API Specification

- Block storage
 - Provisioning with **class of service** control
 - Volume Mapping and Masking
 - Replication
 - Capacity and health metrics
- File system storage
 - Adds File System and File Share
 - Leverages all other concepts – provisioning with class of service, replication, ...
- Additional content
 - Object drive storage

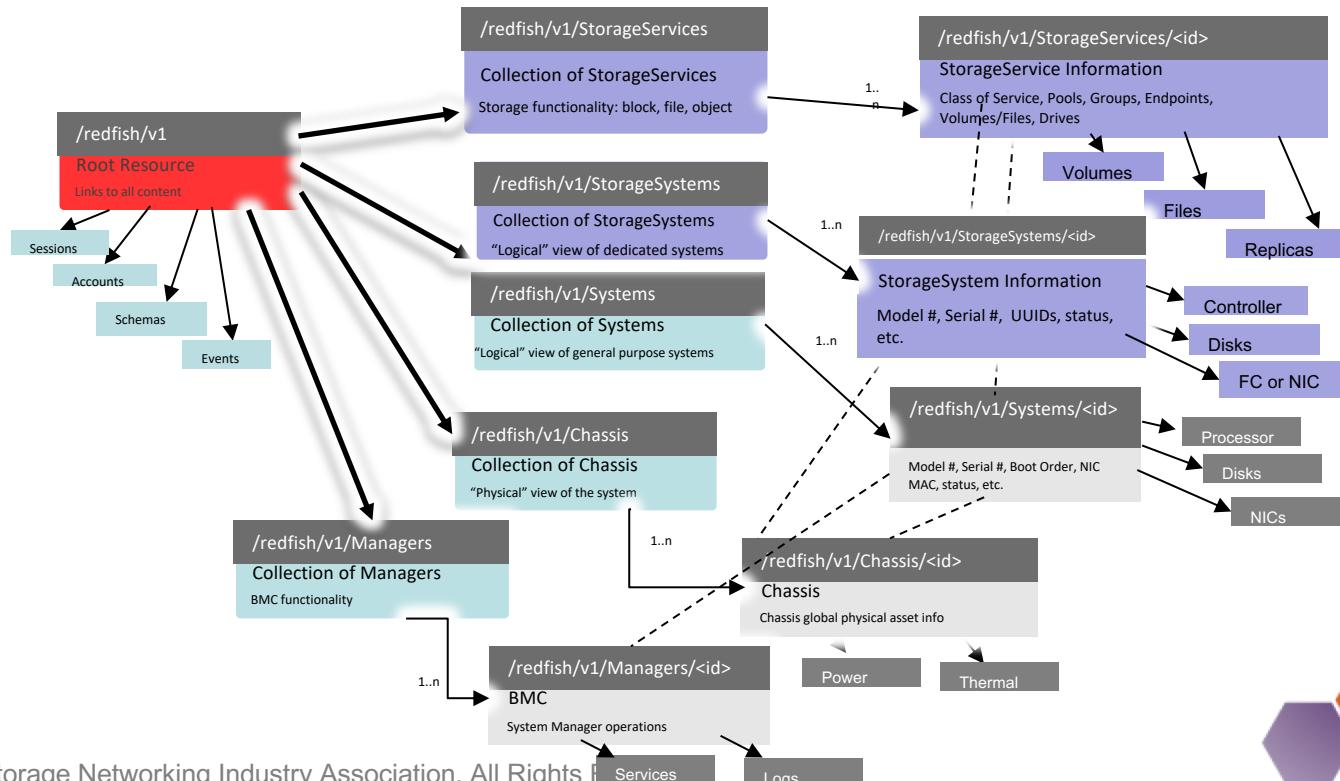


Starting with Redfish: An Overview

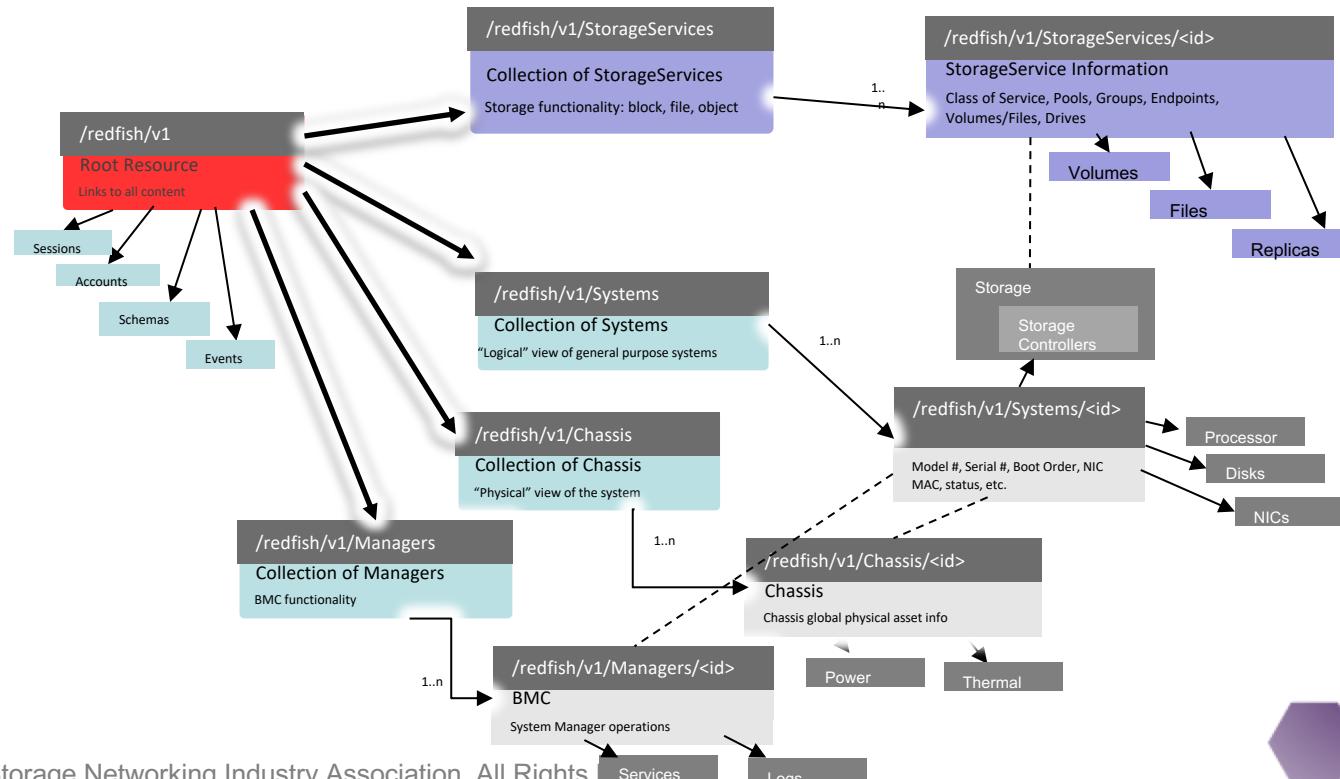
Redfish Resource Map



Adding Storage to Redfish (2 Ways): Hosted Service Configuration



Adding Storage to Redfish (2 Ways): Integrated Service Configuration



See example Swordfish configurations

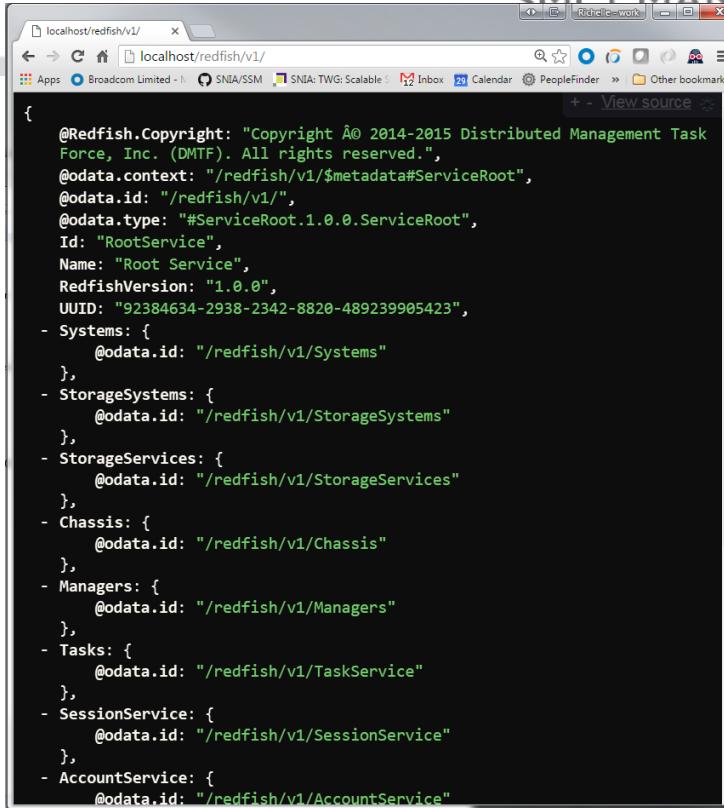
- As a work tool, the Technical Work Group (TWG) works with “mockups” (snapshots of a state in time) of different types of systems
- Published at <http://swordfishmockups.com> (/redfish/v1/)

Note: Mockups are representations of implementations, not normative



Overview of Swordfish

- Explore the Swordfish data model to see potential / typical implementation
- Navigate through the model to learn about and see various resources
- SNIA mockups show two examples of block storage systems
 - Simple: A small external array
 - Complex: all of the elements in the block storage model, with remote replication
- .. and an example of a file server with multiple file shares



A screenshot of a web browser window displaying a JSON API response from the URL `localhost/redfish/v1/`. The browser's address bar shows `localhost/redfish/v1/`. The page title is also `localhost/redfish/v1/`. The page content is a JSON object representing the service root of a Redfish v1 system. The JSON structure includes properties like `@odata.context`, `@odata.id`, `@odata.type`, and a list of resources such as `Systems`, `StorageSystems`, `StorageServices`, `Chassis`, `Managers`, `Tasks`, `SessionService`, and `AccountService`.

```
{
  "@Redfish.Copyright": "Copyright \u00a9 2014-2015 Distributed Management Task Force, Inc. (DMTF). All rights reserved.",
  "@odata.context": "/redfish/v1/$metadata#ServiceRoot",
  "@odata.id": "/redfish/v1/",
  "@odata.type": "#ServiceRoot.1.0.0.ServiceRoot",
  "Id": "RootService",
  "Name": "Root Service",
  "RedfishVersion": "1.0.0",
  "UUID": "92384634-2938-2342-8820-489239905423",
  "- Systems: {
      @odata.id: "/redfish/v1/Systems"
    },
  - StorageSystems: {
      @odata.id: "/redfish/v1/StorageSystems"
    },
  - StorageServices: {
      @odata.id: "/redfish/v1/StorageServices"
    },
  - Chassis: {
      @odata.id: "/redfish/v1/Chassis"
    },
  - Managers: {
      @odata.id: "/redfish/v1/Managers"
    },
  - Tasks: {
      @odata.id: "/redfish/v1/TaskService"
    },
  - SessionService: {
      @odata.id: "/redfish/v1/SessionService"
    },
  - AccountService: {
      @odata.id: "/redfish/v1/AccountService"
    }
}
```

Navigating through the Mockups...

- Select the <..../redfish/v1/StorageServices> link to see the “Collection” of Storage Services
- Click the “<.../StorageServices/Simple>” link to see the details of the Simple mockup or ...
“<.../StorageServices/1>” to see the details of the complex storage service mockup
“<.../StorageServices/FileService>” to see the filesystem mockup



```
localhost/redfish/v1/StorageServices
{
    @Redfish.Copyright: "Copyright 2015-2016 SNIA. All rights reserved.",
    @odata.context: "/redfish/v1/$metadata#StorageService.StorageService",
    @odata.id: "/redfish/v1/StorageSystems/Simple/StorageServices",
    @odata.type: "#StorageServiceCollection.1.0.0.StorageServiceCollection",
    Name: "Storage Service Collection",
    Members@odata.count: 3,
    - Members: [
        - {
            @odata.id: "/redfish/v1/StorageServices/1"
        },
        - {
            @odata.id: "/redfish/v1/StorageServices/FileService"
        },
        - {
            @odata.id: "/redfish/v1/StorageServices/Simple1"
        }
    ]
}
```

What's in a Storage Service? (Block)

- Available Classes Of Service
 - . Lines of Service that are used to compose the Classes of Service
- Volumes
- Pools
- Groups
- Endpoints
- ...
- Pointer to related resources (system, chassis,...)



A screenshot of a web browser window titled "localhost/redfish/v1/StorageService". The address bar shows the URL "localhost/redfish/v1/StorageService". The page content displays a JSON object representing a storage service. The JSON structure includes fields such as @Redfish.Copyright, @odata.context, @odata.id, @odata.type, Id, Name, Description, Status, ClassesOfService, Drives, InitiatorEndpointGroups, TargetEndpointGroups, Endpoints, StorageGroups, StoragePools, Volumes, and Links. The "Endpoints" field is highlighted with a red box.

```
{  
    "@Redfish.Copyright": "Copyright 2014-2016 SNIA. All rights reserved.",  
    "@odata.context": "/redfish/v1/$metadata#StorageService.StorageService",  
    "@odata.id": "/redfish/v1/StorageServices/1",  
    "@odata.type": "#StorageService.1.0.0.StorageService",  
    "Id": "1",  
    "Name": "My Storage Service",  
    "Description": "Description of storage",  
    "+ Status: {...},  
    + ClassesOfService: [...],  
    - Drives: {  
        "@odata.id": "/redfish/v1/Chassis/StorageEnclosure1/Drives"  
    },  
    + InitiatorEndpointGroups: [...],  
    + TargetEndpointGroups: [...],  
    + Endpoints: [...],  
    + StorageGroups: [...],  
    - StoragePools: {  
        "@odata.id": "/redfish/v1/StorageServices/1/StoragePools"  
    },  
    - Volumes: {  
        "@odata.id": "/redfish/v1/StorageServices/1/Volumes"  
    },  
    - Links: {  
        - Enclosures: {  
            "@odata.id": "/redfish/v1/Chassis/1"  
        },  
        - HostingSystem: {  
            "@odata.id": "/redfish/v1/StorageSystems/Complex"  
        },  
        - DataProtectionLoSCapabilities: {  
            "@odata.id":  
                "/redfish/v1/StorageServices/1/DataProtectionLoSCapabilities"  
        },  
        - DataSecurityLoSCapabilities: {  
            "@odata.id":  
                "/redfish/v1/StorageServices/1/DataSecurityLoSCapabilities"  
        }  
    }  
}
```

What's in a Storage Service? (File)

Same structure:

- Available Classes Of Service
- *File systems*
- Pools
- Groups
- Endpoints
- ...
- Pointer to related resources (system, chassis, **block service** or drives)



The screenshot shows a browser window with the URL `localhost/redfish/v1/StorageService`. The page displays a JSON object representing a storage service. The object includes fields for copyright information, context, ID, type, name, description, status (enabled, OK), classes of service, file systems, storage service capabilities, storage groups, storage pools, and links to enclosures and a hosting system. A tooltip for the `StoragePools` field is visible at the bottom of the screen.

```
{  
    "@Redfish.Copyright": "Copyright 2014-2016 SNIA. All rights reserved.",  
    "@odata.context": "/redfish/v1/$metadata#StorageService.StorageService",  
    "@odata.id": "/redfish/v1/StorageServices/FileService",  
    "@odata.type": "#StorageService.1.0.0.StorageService",  
    "Id": "1",  
    "Name": "My Storage Service",  
    "Description": "Description of storage",  
    "Status": {  
        "State": "Enabled",  
        "Health": "OK"  
    },  
    "+ ClassesOfService: [...],  
    "- FileSystems: {  
        "@odata.id": "/redfish/v1/StorageServices/FileService/FileSystems"  
    },  
    - StorageServiceCapabilities: {  
        "@odata.id":  
            "/redfish/v1/StorageServices/FileService/StorageServiceCapabilitie  
    },  
    + StorageGroups: [...],  
    + StoragePools: {...},  
    - Links: {  
        - Enclosures: {  
            "@odata.id": "/redfish/v1/Chassis/1"  
        },  
        - HostingSystem: {  
            "@odata.id": "/redfish/v1/StorageSystems/FileServer"  
        }  
    },  
    "Oem": { }  
}
```

Discovery...

Let's discover something:

Do I have space to...?

1. Check the capacity in a storage pool that I have permission to allocate storage from.
2. Navigate down into “SpecialPool” and check its remaining capacity



A screenshot of a web browser window titled "localhost/redfish/v1/Store". The URL in the address bar is "localhost/redfish/v1/StorageServices/1/StoragePools/SpecialPool". The page content displays a JSON object representing a storage pool. The object includes properties like @SSM.Copyright, @odata.context, @odata.id, @odata.type, Id, Name, Description, BlockSizeBytes, Capacity (with Data, Metadata, and Snapshot fields), CapacitySources (with a single item), and Links (with ClassOfService and ProvidingPool fields). The JSON is color-coded for readability, with green for strings and numbers.

```
{  
    "@SSM.Copyright": "Copyright \u00a9 2014-2016 SNIA. All rights reserved.",  
    "@odata.context": "/redfish/v1/$metadata#StoragePool.StoragePool",  
    "@odata.id": "/redfish/v1/StorageServices/1/StoragePools/SpecialPool",  
    "@odata.type": "#StoragePool_1_0_0.StoragePool",  
    "Id": "SpecialPool",  
    "Name": "SpecialPool",  
    "Description": "Special storage pool",  
    "BlockSizeBytes": 8192,  
    "- Capacity": {  
        "- Data": {  
            "ConsumedBytes": 549755813888,  
            "AllocatedBytes": 1099511627776,  
            "GuaranteedBytes": 70368744177664,  
            "ProvisionedBytes": 140737488355328  
        },  
        "Metadata": null,  
        "Snapshot": null  
    },  
    "- CapacitySources": [  
        {-  
            "- ProvidedCapacity": {  
                "ConsumedBytes": 70368744177664,  
                "AllocatedBytes": 140737488355328,  
                "GuaranteedBytes": 17592186044416,  
                "ProvisionedBytes": 562949953421312  
            },  
            "- Links": {  
                "- ClassOfService": {  
                    "@odata.id":  
                        "/redfish/v1/StorageServices/1/ClassesOfService/Gold"  
                },  
                "- ProvidingPool": {  
                    "@odata.id":  
                        "/redfish/v1/StorageServices/1/StoragePools/BasePool"  
                },  
                "- PoolType": {  
                    "@odata.id":  
                        "/redfish/v1/StorageServices/1/PoolTypes/General"  
                }  
            }  
        }  
    ]  
}
```

How Do Clients Determine Swordfish Implementations They Can Support?



- Profiles define sets of required functionality to support:
 - ◆ Basic Swordfish support
 - > Integrated service configuration
 - > Hosted service configuration
 - ◆ Add-on functionality:
 - > Local replication
 - > Remote replication
 - ◆ Certification / Conformance Requirements
 - > (Planned) EnergyStar Requirements: Orthogonal to functionality profiles
 - Energy and power metrics
 - Controls for on-demand instrumentation



Swordfish Specs and Technical Content... In 2018

- v1.0.6 (WIP) Release in February 2018
 - Introduction of two StorageSystem models
 - Schema updates, Spec section additions, User's guide updates: new use cases for on-demand replicas
 - *Pending release as SNIA Technical Position*
- Work-in-progress:
 - Profile Development: Basic Swordfish Support
- Future Functionality
 - Storage-specific security roles
 - Enhanced Class of Service capabilities for Spare management, rebuild management
 - Enhanced profiles for SNIA Alliance partner organizations
 - Functionality alignment across DMTF, NVMeExpress/NVMe-MI and SNIA
 - Object Storage



Swordfish Growth

- SNIA Scalable Storage Management Technical Work Group (SSM TWG)
 - ◆ (SSM is the group, Swordfish is the Spec)
 - ◆ Scalable Storage Management (SSM) TWG chartered in December 2015
 - ◆ v1.0 Spec Released September 2016
- 2017 Focus: validating spec, initial POC implementations
 - ◆ Swordfish Functionality Enhancements: Specification and Technical Content
 - Releases / Work in progress
 - ◆ Documentation and Supporting Materials
 - ◆ Open Source Tools and Infrastructure Development
 - ◆ Implementation Support
 - Plugfests: https://www.snia.org/forums/smi/tech_programs/lab_program



Documentation and Supporting Materials

- Online Practical Guide
 - [SNIA Swordfish Practical Guide](#)
- NEW! Swordfish School:
 - [Swordfish School Playlist](#) (YouTube)
- Swordfish API Specification
- Webcasts



Pinned repositories

Swordfish-API-Emulator

The Swordfish API Emulator can emulate a Swordfish-based system statically or dynamically. Starting from an initial state described by mock-ups, the emulator can be used to emulate a Swordfish syst...

● Python ★ 3 ⚡ 3

Swordfish-basic-web-client

The Swordfish Basic Web Client can connect to one or more Swordfish services (including the Swordfish emulator), and present in a web UI frame the entire Swordfish hierarchy. The basic web client a...

● TypeScript ★ 2

CDMI

The Reference Implementation for the SNIA Cloud Data Management Interface (CDMI) an ISO standard

● Java ⚡ 9

Swordfish-datadog-sample-dashboard-integration

The Swordfish datadog sample dashboard integration provides a sample dashboard for the datadog monitoring service that can connect to a Swordfish services (including the Swordfish emulator), and pr...

★ 1

Swordfish-powerBI-sample-dashboard-integration

The Swordfish PowerBI sample dashboard integration provides a sample dashboard for the PowerBI data center monitoring system that can connect to a Swordfish service (including the Swordfish emulato...

★ 1

OSL

Open Source Liaison – A multi-vendor collaborative group for aligning open source projects with standards

★ 1



How to Participate: Shaping the Standard

- Find pointers to the latest technical content:
 - <http://snia.org/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>
- 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>





THANK YOU

