Swordfish Scalable Storage Management API Specification

Version 1.0.7a

ABSTRACT: The Swordfish Scalable Storage Management API defines a RESTful interface and a standardized data model to provide a scalable, customer-centric interface for managing storage and related data services.

Technical Position

Last Updated 23 December 2018

USAGE

DISCLAIMER

Revision History

Date	Revision	Notes
19 September 2016	1.0.0	Initial Release
12 October 2016	1.0.1	Errata release for general clean up and formatting consistency
1 November 2016	1.0.2	Errata release to change multiple collections' types from collections (arrays) to ResourceCollections to conform to Redfish usage guidelines Change multiple collections' types from collections (arrays) to ResourceCollections to conform to Redfish usage guidelines and move NavigationProperties from Links section.
24 January 2017	1.0.3	Errata release to move complex types and enum to versioned namespace Schedule schema: add property json schema fix (Swordfish to swordfish) Specification enhancements, multiple areas User's guide: multiple new use cases and new document section

Date	Revision	Notes
25 April 2017	1.0.4	Errata release with minor updates to schema: move FileShare collection, integrate DMTF and SNIA versions of Volume, fix incorrect property references and update descriptions. Update mockups. User's guide: Update cross-references.
3 October 2017	1.0.5	Errata release to include schema simplifications and other lessons from initial implementations, as well as general cleanup of specification.
13 February 2018	1.0.6	Updated Storage Systems model – added notion of Integrated Service Configuration in addition to (and named) Hosted Service Configuration Added ComplexType common definition section Added/updated common Redfish property definitions Updates to conform to new SNIA templates.
12 October 2018	1.0.7	Enhanced Spare Capacity Management Model; Deprecated Remaining Capacity Added OpenAPI support: schema references and OpenAPI YAML files Added iSCSI properties for CHAP Event usage enhancements and guidance Volume schema updates – RAID Type enum (deprecating VolumeType usage), add ReplicaTargets Schema updates: Annotations enhancements: Capabilities designations, owning entities, Redfish.Required usage Clarified and updated ClassOfService IsDefault property usage Updated Capabilities location in hierarchy Fix cardinality issue of StorageReplicaInfo usage in StorageGroups and Volume Consolidate Client and Server Endpoint Groups into single Endpoint Group entity (deprecate usage of separate Client Endpoint Group and Server Endpoint Group) Add MappedVolume construct to StorageGroup – adds LUN info and other properties
8 November 2018	1.0.7a	Restored RAIDType property that was missing from 1.0.7 Minor correction to schema versioning

Current Revision

SNIA is actively engaged in expanding and refining the Swordfish specification. The most current revision can be found on the SNIA web site at https://www.snia.org/tech_activities/standards/curr_standards/swordfish.

Contact SNIA

SNIA Web Site

Current SNIA practice is to make updates and other information available through their web site at http://www.snia.org.

FEEDBACK AND INTERPRETATIONS

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent via the SNIA Feedback Portal at http://www.snia.org/feedback/ or by mail to the Storage Networking Industry Association, 4360 ArrowsWest Drive, Colorado Springs, Colorado 80907, U.S.A.

INTENDED AUDIENCE

This document is intended for use by individuals and companies engaged in storage management.

VERSIONING POLICY

This document is versioned material. Versioned material shall have a three-level revision identifier, comprised of a version number 'v', a release number 'r' and an errata number 'e'. Future publications of this document are subject to specific constraints on the scope of change that is permissible from one revision to the next and the degree of interoperability and backward compatibility that should be assumed between products designed to this standard. This versioning policy applies to all SNIA Swordfish versioned materials.

Version Number: Versioned material having version number 'v' shall be backwards compatible with all of revisions of that material that have the same version number 'v'. There is no assurance of interoperability or backward compatibility between revisions of a versioned material with different version numbers.

Release Number: Versioned material with a version number 'v' and release number 'r' shall be backwards compatible with previous revisions of the material with the same version number, and a lower release number. A minor revision represents a technical change to existing content or an adjustment to the scope of the versioned material. Each minor revision causes the release number to be increased by one.

Errata Number: Versioned material having version number 'v', a release number 'r', and an errata number 'e' should be backwards compatible with previous revisions of the material with the same version number and release number ("errata versions"). An errata revision of versioned material is limited to minor corrections or clarifications of existing versioned material. An errata revision may be backwards incompatible, if the incompatibility is necessary for correct operation of implementations of the versioned material.

About SNIA

The Storage Networking Industry Association (SNIA) is a non-profit organization made up of member companies spanning information technology. A globally recognized and trusted authority, SNIA's mission is to lead the storage industry in developing and promoting vendor-neutral architectures, standards and educational services that facilitate the efficient management, movement and security of information.

Acknowledgements

The SNIA Scalable Storage Management Technical Work Group, which developed and reviewed this work in progress, would like to recognize the significant contributions made by the following members:

Table 2: Contributors

Member	Representatives
Broadcom, Inc.	Richelle Ahlvers
Dell Inc.	Patrick Boyd George Ericson Michael Raineri Rich Roscoe
Hitachi Data Systems	Eric Hibbard
Hewlett Packard Enterprise	Jeff Hilland Chris Lionetti John Mendonca Doug Voigt
Inova Development Inc.	Karl Schopmeyer
Intel Corporation	Slawek Putyrski Paul von Behren
Microsemi	Anand Nagarjan
Microsoft Corporation	Hector Linares Jim Pinkerton Michael Pizzo Scott Seligman
NetApp, Inc.	Don Deel Nilesh Maheshwari
ScienceLogic	Patrick Strick
VMware, Inc.	Murali Rajagopal

Table of Contents

1 Abstract

The Swordfish Scalable Storage Management API ("Swordfish") defines a RESTful interface and a standardized data model to provide a scalable, customer-centric interface for managing storage and related data services. It extends the Redfish Scalable Platforms Management API Specification (DSP0266) from the DMTF.

2 Scope

Swordfish extends the Redfish Scalable Platforms Management API Specification to define a comprehensive, RESTful API for storage management that addresses block storage, file systems, object storage, and storage network infrastructure. It is centered around common operational and business concerns of storage management, including:

- Configuration and provisioning
- Monitoring
- Event and log management
- Performance assessment
- Diagnostics
- Fault detection and remediation
- Security
- Accounting and resource consumption

Swordfish's storage model is built around well-defined classes of service, which provide a means to map high-level business goals and objectives to specific, storage-based actions and requirements, in a clear and consistent way that can be applied uniformly across a broad spectrum of storage configurations and storage types (e.g., block storage, file systems, object stores). Common storage management functionality covered by class of service includes snapshots, replication, mapping and masking, and provisioning.

The Redfish specification provides the protocols and a core set of data models and behaviors for the management of systems. It defines the elements and behaviors that are mandatory for all Redfish implementations. Additionally it defines additional elements and behaviors that can be chosen by system vendors or manufacturers. The specifications also defines points at which OEM (system vendor) extensions can be provided by a given implementation. The specifications specifies normative requirements for Redfish Services and associated materials, such as Redfish Schema files. The Redfish specifications does not set requirements for Redfish clients, but will indicate what a Redfish client should do in order to access and utilize a Redfish Service successfully and effectively.

The Swordfish specification defines additional data models and behaviors for the management of storage systems and storage infrastructure. A Swordfish implementation shall conform to all requirements specified in the Redfish specifications.

Swordfish is suitable for a wide range of storage, from small-scale object drives, integrated RAID cards or RBODs

providing storage services, to external disk arrays or file servers, to infrastructure providing storage services for converged, hyperscale and large scale cloud environments.

This document defines the Swordfish Scalable Storage Management API.

3 Normative References

3.1 Overview

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

3.2 Approved references

Table 1: Approved normative references

Tag	Title (Version)	Author	URL
ISO-8601	Data elements and interchange formats -- Information interchange -- Representation of dates and times -- Part 1: Basic rules	ISO/IEC	http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=70907
ISO-Direct	ISO/IEC Directives, Part 2 Principles and rules for the structure and drafting of ISO and IEC documents (Seventh Edition, 2016)	ISO/IEC	http://isotc.iso.org/livelink/livelink/fetch/2000/2122/4230450/4230456/ISO_IEC_Directives *Part_2 Principles_and_rules_for_the structure_and_drafting_of_ISO_and_IEC* documents_-*2016*%287th_edition%29_-_PDF.pdf?nodeid=17667902&vernum=-2

Tag	Title (Version)	Author	URL
Redfish	Redfish Scalable Platforms Management API Specification (v1.4.0)	DMTF	http://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.4.0.pdf
OData	Open Data Protocol (v. 4.0)	OASIS	https://www.oasis-open.org/standards#odatav4.0
RFC3986	Uniform Resource Identifier (URI): Generic Syntax (2005)	The Internet Society	http://www.rfc-base.org/txt/rfc-3986.txt
CSDL	Common Schema Definition Language (4.0)	OASIS	http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part3-csdl.html
ITIL	ITIL Glossary (2011)	ITIL	https://www.axelos.com/Corporate/media/Files/Glossaries/ITIL_2011_Glossary_GB-v1-0.pdf
Units	The Unified Code for Units of Measure (v2.0.1)	Regenstrief Institute, Inc. and the UCUM Organization	http://unitsofmeasure.org/trac
TLS	Transport Layer Security (TLS) Protocol Version 1.2	IETF	https://www.ietf.org/rfc/rfc5246.txt
SPC-4	SCSI Primary Commands - 4 (SPC-4) INCITS 513-2015	T10	http://www.techstreet.com/cgi-bin/joint.cgi/incits

3.3 References under development

Documents referenced in this section are under active development, and subject to revision or replacement at any time. In the event that the provided URL is no longer valid, refer to the related parent page to locate a replacement.

Table 2: References under development

Tag	Title (Version)	Author	URL	Parent Page
RedfishResource	Redfish Resource and Schema Guide	DMTF	http://www.dmtf.org/sites/default/files/standards/documents/DSP2046_2017.0a.pdf	http://www.dmtf.org/redfish

3.4 Other references

None defined in this document.

4 Terms and Definitions

4.1 Overview

In this document, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause. New terms, frequently used Redfish terms.

4.2 Swordfish-specific Terms

4.2.1 Definitions

The following terms are used in this document.

Table 3: Swordfish terms

Term	Definition
Entity	An element in a model that represents resources. The element may be either a type declaration or a model instance representing an instance of the resource.
Entity Instance	A model element that represents the information and behaviors of a particular instance of an entity.
Entity Type	A model element that specifies the structure, information and behaviors of an entity.
Instance	See Entity Instance.

Term	Definition
OData service	A REST-based service that allows resources, identified using Uniform Resource Locators (URLs) and defined in a model, to be published and edited by Web clients using simple HTTP messages.
Metamodel	A model that defines the semantics for the construction of a model.
Model	A set of entities and the relationships between them that define the semantics, behavior and state of that set.
Resource	A named item of interest. The item may be be a collection of other items. A resource may be assigned a URI that allows it to receive and process messages. A particular instance of a resource is represented in the model by an entity instance. The type of a resource is represented in the model by an entity type.
Schema	A formal language representation of a model that conforms to a metamodel.
Service Document	The term Service Document is used to refer to a particular resource that is directly accessed via the OData service entry point. This resource serves as a starting point for locating and accessing the other resources and associated metadata that together make up an instance of a Swordfish Service. See also OData Service Document
Swordfish service	A service that is a Redfish service and that implements Swordfish extensions to the Redfish model that conform to the requirements of this document.

4.2.2 Symbols and abbreviated terms

None in this document.

4.3 Reference to Redfish terms

Many terms in this document were originally defined in the Redfish Specification. Some of the more common terms and definitions are reproduced here, as an aid to the reader.

Table 4: Redfish terms

Term	Definition (as of 24 January 2017)
OData	The Open Data Protocol, as defined in OData-Protocol.
OData Service Document	The name for a resource that provides information about the Service Root. The Service Document provides a standard format for enumerating the resources exposed by the service that enables generic hypermedia-driven OData clients to navigate to the resources of the Redfish Service. See also Service Document
Redfish Schema	The CSDL defintion of Redfish resources.
Redfish service	An OData service that conforms to requirements of the Redfish specification.

Term	Definition (as of 24 January 2017)
Redfish Service Entry Point	Also referred to as "Service Entry Point". An URI through which a particular instance of a Redfish Service is accessed. A Redfish Service may have more than one Service Entry Point
Request	A message from a Client to a Server. It consists of a request line (which includes the Operation), request headers, an empty line and an optional message body.
Service Root	The term Service Root is used to refer to a particular resource that is directly accessed via the Redfish service entry point. This resource serves as a starting point for locating and accessing the other resources and associated metadata that together make up an instance of a Redfish Service.

4.4 Keywords (normative language terms)

This document conforms to ISO/IEC Directives, Part 2 for keyword usage. The most common terms and their intended meanings are summarized below.

Table 5: Normative language terms

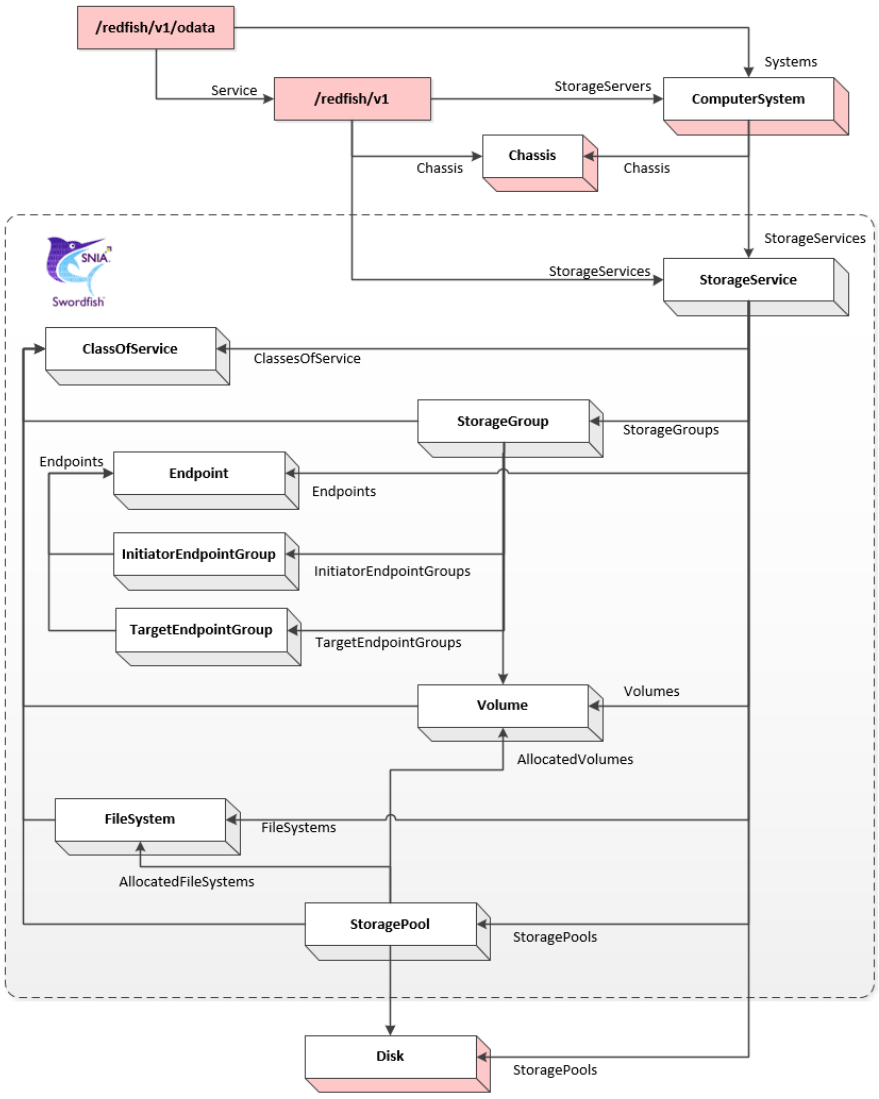
Term(s)	Meaning
shall / shall not	Used to identify objectively verifiable criteria to be fulfilled and from which no deviation is permitted if compliance with the document is to be claimed
should / should not	Used to identify a suggested possible choice or course of action deemed to be particularly suitable without necessarily mentioning or excluding others
may / need not	Used to convey consent or liberty (or opportunity) to do something
can / cannot	Expected or conceivable material, physical or causal outcome
must	Identifies a constraint or obligation on the user of the document, typically due to one or more legal requirements or laws of nature, that is not stated as a provision of the standard *NB:* "must" is not an alternative for "shall", and should only be used for constraints that arise from outside this standard

5 Swordfish Overview

5.1 Introduction

The Swordfish Scalable Storage Management API ("Swordfish") defines a RESTful interface and a standardized data model to provide a scalable, customer-centric interface for managing storage and related data services. It extends the Redfish Scalable Platforms Management API Specification (DSP0266) from the DMTF.

5.2 Relation to Redfish



The Swordfish service interface extends the Redfish service interface. As such, a Swordfish service is a Redfish service and includes all required elements of the Redfish model.

Storage systems managed by the Swordfish storage service are located in the `ServiceRoot` (and `ServiceContainer`) via the `StorageSystems` resource collection. They are modeled using Redfish `ComputerSystems`. The physical infrastructure is modeled using Redfish `Chassis`.

Each Swordfish `StorageService` is located in the `ServiceRoot` (and `ServiceContainer`) via the `StorageServices` resource collection. All Swordfish defined instances are located through `StorageService` intances. A Swordfish management client may focus entirely on entities defined by the Swordfish schema.

The combined Redfish and Swordfish models defines information requirements and constraints on the values that are

used as input or output of the operations supported by the Swordfish interface. The Swordfish interface relies on the operations specified by the OData REST protocol (#normative-references). Additional operations (known as Actions) are also defined by the model. The information content is defined by a schema specified using the Common Schema Definition Language (CSDL) (#normative-references) defined by the OData organization within OASIS (https://www.oasis-open.org/).

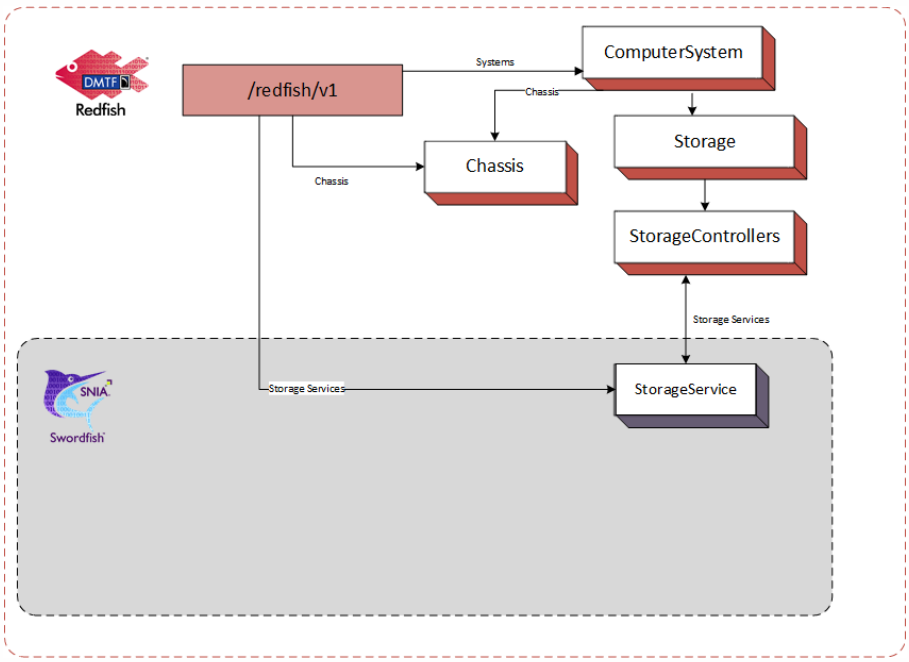
Each Swordfish service is accessed via well known URLs on the system supporting the Swordfish Service. Since Swordfish is an extension of Redfish, these URLs are the same as for accessing the Redfish defined aspects of the service.

5.2.1 Storage System Models

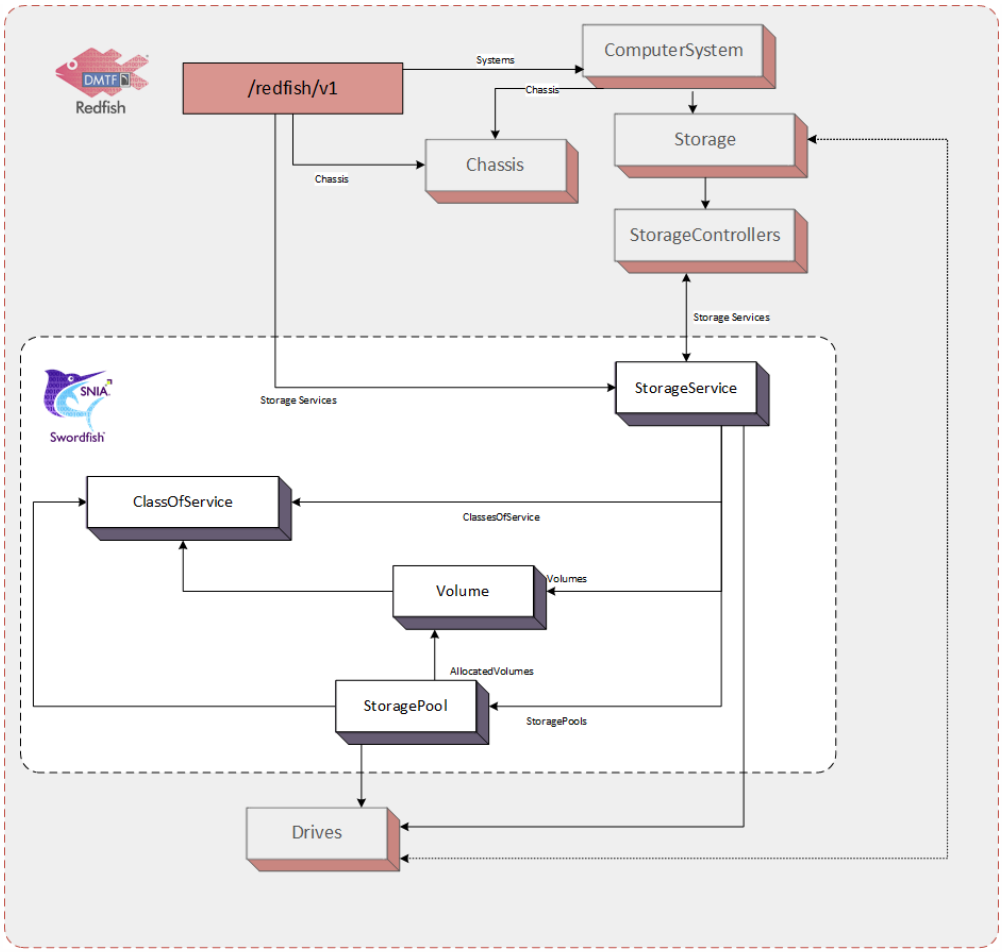
Swordfish has been designed to support a broad range of configurations, requirements, size and complexity, as well as logical and physical architectures. As a result, there are two primary methods of modelling the storage system for a Swordfish implementation. Either model choice results in the same storage service, regardless of the storage system model.

1. Integrated Service Configuration

The storage systems managed by the Swordfish storage service are modeled using the Redfish `Storage` resource and `StorageController` resource collections. The `Storage` resource is located in the Redfish hierarchy contained by `ComputerSystems`, typically running as ApplicationServers. The physical infrastructure is modeled using Redfish `Chassis`.

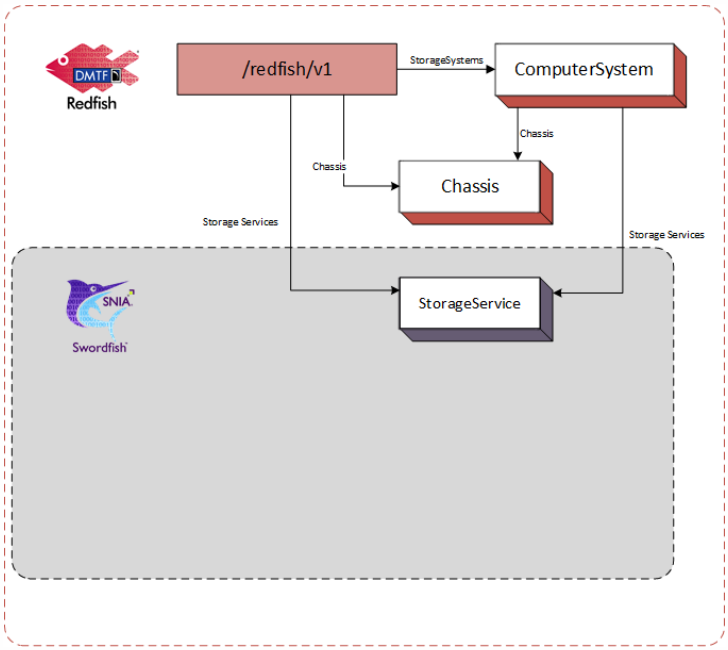


This configuration works well when the storage service is hosted by a storage resource within a computer system. An example of a Storage Service for an integrated service configuration is shown below.



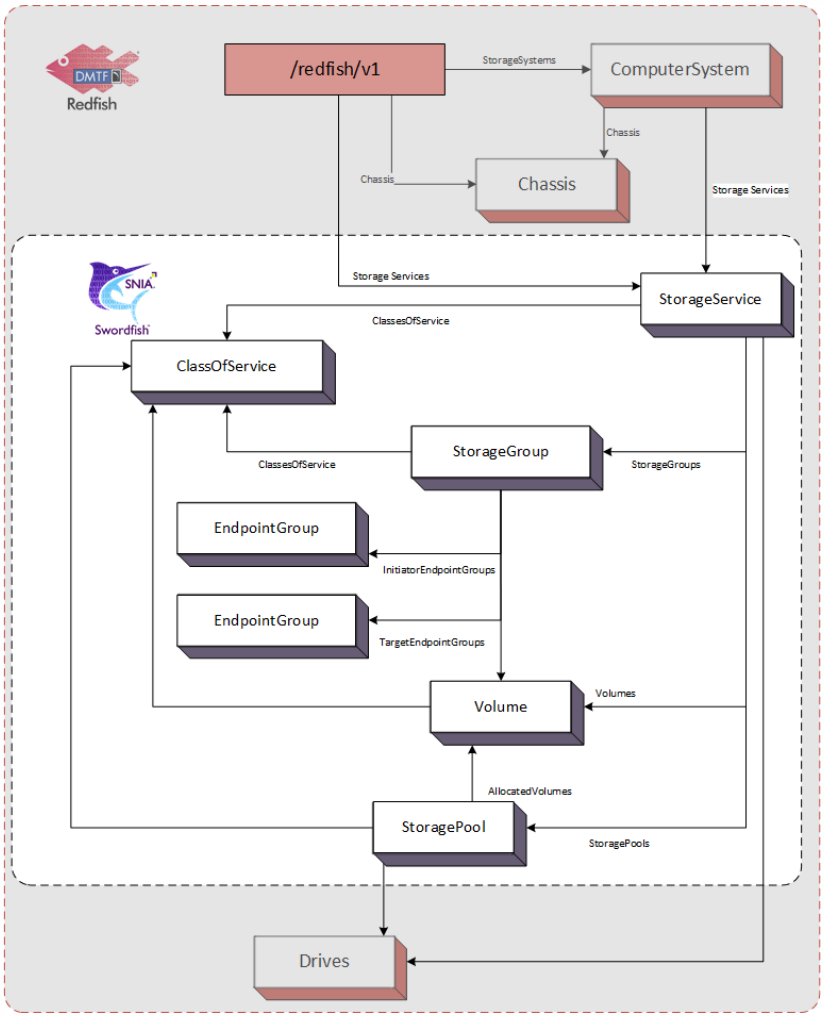
2. Hosted Service Configuration

The storage systems managed by the Swordfish storage service are located in the `ServiceRoot` (and `ServiceContainer`) via the `StorageSystems` resource collection. They are modeled using Redfish `ComputerSystems`. The physical infrastructure is modeled using Redfish `Chassis`.



This configuration works well when the storage system hosts the storage service directly. An example of a Storage Service

for a hosted service configuration is shown below.



5.2.2 The ServiceRoot and ServiceContainer entities

5.2.2.1 Overview

A **GET** of `/redfish/v1` will return the `ServiceRoot` entity. A **GET** of `/redfish/v1/odata` will return the `ServiceContainer` instances that represents the OData service document. Each of these instances provides links to the remainder of the system.

The following are the elements utilized for Swordfish management.

- `Systems`: A reference to a Systems resource collection;
- `Chassis`: A reference to a Chasis resource collection;
- `StorageSystems`: A reference to a StorageSystems resource collection;
- `StorageServices`: A reference to a StorageServices resource collection.

5.2.2.2 The Systems resource collection

A resource collection that references a set of `ComputerSystem` resources that each represents a general purpose

application server. Each `ComputerSystem` resource will have an entry with the value of "ApplicationServer" in its `HostingRoles` property. A particular `ComputerSystem` resource can be in both the StorageSystems collection and the Systems collection.

5.2.2.3 The Chassis resource collection

A resource collection that references a set of `Chassis` resources. Each Chassis resource represents physical containers, (i.e. sheet-metal confined spaces and logical zones like racks, enclosures, chassis and all other containers). Subsystems (like sensors), which operate outside of a system's data plane (meaning the resources are not accessible to software running on the system) are linked either directly or indirectly through this resource.

5.2.2.4 The StorageSystems resource collection

A reference to a `ComputerSystemCollection` with members of type `ComputerSystem` that support storage services. These `ComputerSystem` resources represent systems that support Swordfish storage management services. They will have an entry with the value of "StorageServer" in their `HostingRoles` property.A resource collection that references a set of `ComputerSystem` resources that each represents a storage server. Each `ComputerSystem` resource will have an entry with the value of "StorageServer" in its `HostingRoles` property. A particular `ComputerSystem` resource can be a member of both the `StorageSystems` resource collection and the `Systems` resource collection.

5.2.2.5 The StorageServices resource collection

A reference to a `StorageServiceCollection` with members that are of type `StorageService`. A resource collection that references a set of `StorageService` resources. Each `StorageService` resource represents the resources and behaviors supported by that storage service.

5.3 Storage Services

5.3.1 The StorageService resource

The storage service is hosted on a storage system and exposes logical storage, associated resources and related functionality. Storage service resources can be found in the service root or service container via the `StorageServices` resource collection.

The following are the principal properties of `StorageService` that point to resources managed or defined by the storage service:

- `ClassesOfService`: A reference to a resource collection that specifies the supported `ClassOfService` resources.
- `ClientEndpointGroups`: A reference to a resource collection that collects `ClientEndpointGroup` resources.
- `Drives`: A reference to a resource collection that collects `Drive` resources used for storage.
- `Enclosures`: A reference to a resource collection that collects Chassis resources that contain storage related resources.

- `Endponts`: A reference to a resource collection that collects`Endpoint` resources used to access storage.
- `FileSystems`: A reference to a resource collection that collects `FileSystem` resources.
- `ServerEndpointGroups`: A reference to a resource collection that collects `ServerEndpointGroup` resources.
- `StorageGroups`: A reference to a resource collection that collects `StorageGroup` resources.
- `StoragePools`: A reference to a resource collection that collects `StorageGroup` resources.
- `Volumes`: A reference to a resource collection that collects `Volume` resources.
- `HostingSystem`: A reference to the `ComputerSystem` instance that hosts this `StorageService`.

The following properties each include a set of attributes that each describe a range of capabilities that the storage service can support for a particular kind of service.

- `DataProtectionLoSCapabilities`: Replicas that protects data from loss.
- `DataSecurityLoSCapabilities`: Data security service level requirements. The data security characteristics enable the storage system to be used in an environment where compliance with an externally-specified security standard or standards is required. Examples of such standards include FIPS-140, HIPAA and PCI.
- `DataStorageLoSCapabilities`: Provisioning and access characteristics for storage of the data.
- `IOConnectivityLoSCapabilities`: IO connectivity requirements for access to the data.
- `IOPerformanceLoSCapabilities`: IO performance requirements for access to the data.

In each of the above, not all combinations of attribute values are likely to be supported by the storage service.

Known supported combinations of attribute values are used to construct entries in the `LinesOfService` array property. Not all attributes of a line of service entry need be specified (i.e. some may be Null). If an attribute has no value, the storage service may choose any supported values when provisioning for that entry. Otherwise, the line of service attribute values specifies the kind or level of service to be provided.

5.4 The ClassOfService resource

A class of service represents a choice of utility or warranty offered to customers by a service. (ITIL uses the term `service option`. See the Normative References.)

Each `ClassOfService` resource is a uniquely named description of the characteristics of one choice of utility or warranty for a service. Each `ClassOfService` is a description of the kind and quality of service to provide and is not intended to describe how the service provides that service.

Each `ClassOfService` is defined by an aggregation of lines of service. Supported lines of service are listed in the corresponding capabilities attributes of the storage service, (see above).

Currently defined lines of service are:

- Data Protection: Describes the characteristics of a replica that protects data from loss.
- Data Security: Describe data security service level requirements. The data security characteristics enable the storage system to be used in an environment where compliance with an externally-specified security standard or standards is required. Examples of such standards include FIPS-140, HIPAA and PCI.
- Data Storage: Describes provisioning and access characteristics for storage of the data.
- IO Connectivity: Describes IO connectivity requirements for access to the data.

- IO Performance: Describes the IO performance requirements for access to the data under a particular workload.

Some advertised `ClassOfService` resources are created by the service implementation. These are generally not changeable and are intrinsic to the implementation.

A service may support creation or modification of `ClassOfService` resources. All must be consistent with the capabilities of the service.

5.5 The Endpoint resource

Endpoints represent one end of a protocol specific connection that supports sending or receiving messages according to a particular protocol.

5.6 The Endpoint Collection resource

The `Endpoint` Group is resource collection that references a set of `Endpoint` resources.

5.7 The EndpointGroup resource

The `EndpointGroup` is a resource that represents a set of `Endpoint` resources that have the same management characteristics and which will all have the same access state.

5.8 The EndpointGroupCollection resource

The `EndpointGroupCollection` is resource collection that references a set of `EndpointGroup` resources.

5.9 The StorageGroup resource

StorageGroups represent a set of volumes that are managed as a group with the same consistency requirements. The volumes of a storage group are collectively exposed or hidden to a set of clients.

The set of volumes is specified by the `Volumes` attribute, which is a resource collection that references volumes.

The set of client endpoints to which the volumes can be exposed is specified by the `ClientEndpointGroups`attribute. The `ClientEndpointGroup` resource specifies a collection of `EndpointGroup` resources.

The set of server endpoints to which the volumes can be exposed is specified by the `ServerEndpointGroups`attribute. The `ServerEndpointGroup` resource specifies a collection of `EndpointGroup` resources.

5.10 The StoragePool resource

The `StoragePool` resource represents unassigned storage capacity that can be used to produce storage volumes or other storage pools, which conform to one or more classes of service.

The following are the principal properties of `StoragePool` that are used to identify resources provisioned or supported by the storage pool:

- `ClassesOfService`: A reference to a resource collection that specifies the set `ClassOfService` resources that can be specified when provisioning resources from the storage pool.
- `AllocatedVolumes`: A reference to a resource collection that collects `Volume` resources that have been provisioned from the storage pool.
- `AllocatedPools`: A reference to a resource collection that collects `StoragePool` resources that have been provisioned from the storage pool.
- `DefaultClassOfService`: A reference to the default `ClassOfService` resources used for provisioning from the storage pool.

5.11 The Volume resource

`Volume` resource represents a block-addressable container of storage, sometimes referred to as a "Logical Unit", "LU", "LUN", or "StorageVolume" in the storage industry. Volumes optionally adhere to a `ClassOfService`, which defines added functionality. Examples include:

- Access capabilities
- Capacity and capacity sources
- Consumption tracking (e.g., LowSpaceWarningThresholdPercents)
- Replication details
- StorageGroup Information

5.12 The FileSystem resource

This `FileSystem` resource represents a file system. File systems represent file-addressable capacity that are conformant to a `ClassOfService`. Each `FileSystem` may contain a collection of `FileShares` that can be presented to hosts.

6 Data model

6.1 Swordfish extensions to Redfish

6.1.1 Overview

Redfish has added two properties to the ServiceRoot that provide access to Swordfish resources.

The first is `StorageSystems`. This property references a collection of `ComputerSystem` resources that each support Swordfish functionality. Each such `ComputerSystem` shall have:

- an entry in its `HostingRoles` property with the value of `StorageServer`
- at least one entry in its `StorageServices.Members` property.

The second is `StorageServices`. This property references a collection of `StorageService` resources. It provides the client an efficient means to search across all `StorageService` resources, regardless of which `ComputerSystem` is supporting the service.

6.1.2 Swordfish and Redfish specific OEM or vendor extensions

The Swordfish and Redfish models are extended by subclassing the `OEM` `ComplexTypes` that are defined in the Swordfish and Redfish schemas.

6.1.3 OData specific OEM or vendor extensions

In addition to extending the Redfish model as described above. An OEM may extend the Redfish `ServiceContainer` by defining a new `EntityContainer` that extends the ServiceContainer found in the Redfish ServiceRoot_v1.xml file, (see OData EntityContainer).

> Note: This has the same semantics as subclassing in a typical object oriented environment.

An OEM extended implementation of the Swordfish service would access OEM extensions to `EntityContainer` via the service entry-point `/redfish/v1/odata`.

6.2 Entity Sets

The Swordfish model does not currently expose any explicitly defined entity sets. OData specifies that an entity set is defined for each NavigationProperty that is defined as a collection and that has the ContainsTarget attribute set to true. In all other cases, Swordfish assumes that an entity set is defined globally within the implementation for each entity type. This is effectively the same as if the entity sets were explicitly defined in the ServiceRoot entity container.

6.3 Addressing entities within a collection

An instance (entity) of an EntityType is uniquely identified within its entity set by its key. The URI for the reference may specify the key using one of two general strategies

1. OData recommends specifying the key value within parenthesis following the path segment that identifies the referencing entity set. (See clause "Canonical URL" in in OData)
2. Redfish common practice is to use an alternative form that adds a path segment having the value of the key following the path segment that identifies the referencing collection. (See clause "Alternate Key-as-Segment Syntax" in OData.)

A Swordfish implementation shall support both strategies.

6.4 Addressing members of a ResourceCollection

Redfish specifies that subclasses of ResourceCollection shall include a Members collection property (See clause "Collection resource response" in DSP0266)

Redfish allows a POST request to a ResourceCollection to be equivalent to the same POST request to the Members property of that ResourceCollection.
For a particular ResourceCollection, if a Swordfish implementation supports either form, it shall support both.

It is common practice in Redfish to also eliminate the Members property from any request URI that navigates through a type hierarchy that includes a Member within a ResourceCollection. Care should be taken when defining and using a ResourceCollection subclass to not introduce ambiguities when an explicit reference to a Members property is dropped from a request URI.

6.5 Schema repository

The primary online source for the Swordfish schema shall be co-located on the DMTF schema site with the Redfish schema: http://redfish.dmtf.org/schemas/swordfish Developers may also download the schema as part of the Swordfish bundle from snia.org (refer to snia.org/swordfish for pointers to the bundle locations).

Implementations should refer either to the versions available on the dmtf.org site or to locally provided instances of the schema.

6.6 Referencing other schemas

Swordfish directly reference the following Redfish schemas. - Chassis - ChassisCollection - ComputerSystem - ComputerSystemCollection - Drive - Endpoint - EthernetInterface - EventService - Location - RedfishExtensions - Redundancy - ResourceTask - Schedule - ServiceContainer - ServiceRoot

Other Redfish schema may be added by inference or directly to implementations. Examples are available in the Swordfish mockups.

7 Schema Considerations

7.1 Schema Introduction and Overview

A Swordfish implementation is a Redfish implementation, as such it minimally includes support for some Redfish-defined schema, including ServiceRoot and ComputerSystem. Swordfish implementations include support for Swordfish-defined schema and shall include support for StorageService. A Swordfish implementation is not required to support all of the various hardware oriented schema typically found in a Redfish implementation.

Swordfish schema is conformant with the rules used to define Redfish schema. Redfish schema is conformant with the Common Schema Definition Language, see CSDL. This section provides additional definition and context for the CSDL elements used to define Swordfish schema.

7.2 Common schema attributes

The following table lists common schema attributes used in the definition of Swordfish, for details see CSDL

Table 6: Schema attributes

Name	Applies to	Description
Abstract	ComplexType, EntityType	If true, the entity may not be instantiated
BaseType	ComplexType, EntityType	Names an inherited element.
DefaultValue	Property	The value of a property if not explicitly set
Name	All	The name of the schema element
Nullable	NavigationProperty, Property	If false, the qualified property shall have a value. The default value is true. A navigation property whose Type attribute specifies a collection shall not specify Nullable=false, as the collection always exists, but may just be empty. *Note: Null is not itself a value, but is an indication of no value.*
Type	Property	The type of the element

7.3 Default values and NULLABLE attributes

The interaction of `Nullable` and `DefaultValue` needs to be clearly understood by both implementers and client developers. The possible combinations of are summarized in Table 6. The table contains:

- **Nullable**: True, if a given property may be NULL

- **DefaultValue**: True, if a default value is provided for a given property
- **Client**: True, if a client value is supplied for a given property in a query or response
- **Result**: The resultant value of the given property. One of:
  - *C*: The client-provided value
  - *D*: The default value
  - *Null*: Null
  - *I*: Implementation defined
  - *Error*: Error state

Table 7: Default and Nullable Interaction

Nullable	DefaultValue	Client	Value
T	T	T	C
T	T	F	D
T	F	T	C
T	F	F	I or Null
F	T	T	C
F	T	F	D
F	F	T	C
F	F	F	I or Error

7.4 Common schema annotations

The following table lists common annotation used in the definition of Swordfish, for details see OData Capabilities Vocabulary, OData Core Vocabulary, OData Measures Vocabulary, and Redfish Extensions,

Table 8: Schema annotations

Name	Applies to	Description
AllowableValues	Parameter	The set of allowable values for a parameter
AutoExpand	NavigationProperty	If true, return expand the target element
AutoExpandReferences	NavigationProperty	If true, return references to the target element
ConformanceLevel	EntityContainer	Specifies OData conformance level
Deprecated	All	Specifies that the element may be removed in future major revisions, but shall continue to be supported as specified in the current revision.
Description	All	A brief description of a model element
LongDescription	All	A normative description of a model element
Maximum	Parameter, Property	Maximum value that an integer property or parameter may have
Minimum	Parameter, Property	Minimum value that an integer property or parameter may have

Name	Applies to	Description
Pattern	Parameter, Property	Specifies a pattern that the value shall match
Permissions	NavigationProperty, Property	Access permission for the property.
Required	NavigationProperty, Property	If true, property is required to be supported by the service. The default is optional. See *Required Properties*
RequiredOnCreate	NavigationProperty, Property	If true, property is required on creation. See *Required Properties*
Unit	Property	The unit of measure for the value.

7.5 Property implementation requirements

The client and the implementer should understand that, regardless of the schema declaration, an implementer may choose to not implement a property. If not implemented, a representation of the property will not be present in a reply. This should not be confused with a response that indicates that a property has been implemented, but has no value (i.e. *propertyName = null*).

There are several factors that could affect the implementation choice. Implementation requirements can be defined in many documents. At a minimum, a developer should review: - this document, - the Redfish specification, and - associated profile specifications.

If a property is implemented, and its schema definition includes the *Redfish.Required* annotation, the property and its current value shall be returned in response to a GET operation against any associated services. If the value of the property is not available, then an implementation shall return *NULL* in place of the value.

If a property is implemented, and its schema definitions includes the *Redfish.RequiredOnCreate* annotation, the property and an appropriate value shall be included in any creation (i.e., POST or PATCH) operation against associated services.

7.6 Schema repository

The primary online source for the Swordfish schema shall be co-located on the DMTF schema site with the Redfish schema: http://redfish.dmtf.org/schemas/swordfish Developers may also download the schema as part of the Swordfish bundle from snia.org (refer to snia.org/swordfish for pointers to the bundle locations).

Implementations should refer either to the versions available on the dmtf.org site or to locally provided instances of the schema.

7.7 Referencing other schemas

Swordfish directly references the following Redfish schemas:

Table 9: Referenced Redfish Schema

Redfish Schema
Chassis
ChassisCollection
ComputerSystem
ComputerSystemCollection
Drive
Endpoint
EthernetInterface
EventService
Location
RedfishExtensions
Redundancy
ResourceTask
Schedule
ServiceContainer
ServiceRoot

Other Redfish schema may be added by inference or directly to implementations. Examples are available in the Swordfish mockups.

8 Implementation requirements

8.1 Security

This document generally adheres to the security requirements defined in the Redfish Specification. It extends the Redfish security model in one important way:

- Swordfish implementations shall implement TLS version 1.2 or greater.

8.2 General constraints

8.2.1 Redfish elements

The Swordfish service interface extends the Redfish service interface. As such, a Swordfish service is a Redfish service and all required elements of the Redfish model shall be present in a Swordfish model.

Swordfish functionality shall not conflict with any previously defined Redfish functionality but it may add to or extend it, and it may add additional constraints on Redfish functionality.

Additionally, any functionality desired in a Swordfish implementation that is specified in Redfish shall follow the requirements as specified in the Redfish specification.

8.2.2 Storage Events

8.2.2.1 Overview:

A Swordfish implementation should implement an event service. Redfish defines the Event Service framework, client subscription model, event delivery mechanism, as well as standard message registries. Swordfish extends the standard message registries to provide additional message registries that correspond to Swordfish-specific services and properties.

The Redfish event service publishes a list of event types supported, and maintains a list of clients that have subscribed. Each subscription maps clients, subscribed events, and the resources that generate them.

8.2.2.2 Message Registry Selection and Management

Swordfish constrains the existing event model to provide a more consistent handling of event notifications and the related messages, in order to assure that client systems can easily and consistently parse and respond to system-level events.

8.2.2.3 Required Usage

- The Resource Event Message Registry defines the underlying messaging model, and shall be used to map messages to resources for storage implementations.
- The Redfish Base Message Registry shall be used to support HTTP connection/error/protocol issues, and general errors.
- The Swordfish Message Registry shall be used as a supplement for the resource event message registry.
- If the Swordfish service implements Redfish tasks (i.e., long-running operations), the implementation shall use the messages defined in the Task Event Message Registry to report status.

8.2.2.4 Recommended Usage

- Standard Messages should be used, wherever possible.
- OEM messages should be avoided. Suggestions for clarification or expansion of the existing registries are encouraged. (submissions should be sent to the SNIA Feedback Portal)

8.3 Discovering Swordfish resources

Each Swordfish implementation supports the following well-known URLs, as defined in Redfish. Specifically:

- `/Redfish`, which contains one or more version properties for the integrated Swordfish and Redfish implementation, starting with v1.
- `/Redfish/v1`, which addresses a `ServiceRoot` instance, which defines the Redfish default principal starting information for version 1 implementation of an integrated Redfish and Swordfish service. A GET operation to it shall retrieve the value of an instance of a `ServiceRoot` `EntityType` as defined in the ServiceRoot_v1.xml file.
- `/Redfish/v1/odata`, which addresses a `ServiceContainer` instance, which defines OData conformant principal starting information for the same version 1 implementation of an integrated Redfish and Swordfish service. A GET operation shall retrieve the value of an instance of a `ServiceContainer` `EntityContainer` as defined in the ServiceRoot_v1.xml file.

> Note: Since the `ServiceContainer` is required to return an `@odata.context` value of `/redfish/v1`, all other elements accessed via it will be the same elements found via the `ServiceRoot`.

> Note: A Swordfish service is a Redfish service with extensions to support storage management. No additional service entry-points are necessary.

Both the `ServiceRoot` and `ServiceContainer` contain a resource collection named `Systems` that lists `ComputerSystem` instances. A `ComputerSystem` instance that supports Swordfish defined services will have a value of "StorageServer" in an entry of its `HostingRoles` property.

The `ServiceContainer` additionally has a Service attribute that references the `ServiceRoot` resource.

Regardless of starting point, the property values of the `ServiceRoot` instance enable navigation to all other resources exposed by the Swordfish service.

8.4 ClassOfService requirements

Each `ClassOfService` shall include at least one line of service. The providing server shall assure that the line of service values of a `ClassOfService` collectively represent a supported choice of service.

8.5 StorageSystems requirements

For Hosted Service Configurations, this property of the `ServiceRoot` references a collection of `ComputerSystem` resources that each support Swordfish functionality. Each `ComputerSystem` included in the `StorageSystems` entry in the `ServiceRoot` shall have:

- an entry in its `HostingRoles` property with the value of `StorageServer`
- at least one entry in its `StorageServices.Members` property.

For Integrated Service Configurations, the StorageSystems concept is realized through the `StorageController` resource. Each `StorageController` instantiated as a Swordfish StorageSystem shall have:

- at least one entry in its `StorageController.Links` property `StorageServices` collection identifying related `StorageServices`

8.6 Entity Sets

The Swordfish model does not currently expose any explicitly defined entity sets. OData specifies that an entity set is defined for each NavigationProperty that is defined as a collection and that has the ContainsTarget attribute set to true. In all other cases, Swordfish assumes that an entity set is defined globally within the implementation for each entity type. This is effectively the same as if the entity sets were explicitly defined in the ServiceRoot entity container.

8.7 Addressing entities within a collection

An instance (entity) of an EntityType is uniquely identified within its entity set by its key. The URI for the reference may specify the key using one of two general strategies

1. OData recommends specifying the key value within parenthesis following the path segment that identifies the referencing entity set. (See clause "Canonical URL" in in OData)
2. Redfish common practice is to use an alternative form that adds a path segment having the value of the key following the path segment that identifies the referencing collection. (See clause "Alternate Key-as-Segment Syntax" in OData.)

A Swordfish implementation shall support both strategies.

8.8 Addressing members of a ResourceCollection

Redfish specifies that subclasses of ResourceCollection shall include a Members collection property (See clause "Collection resource response" in DSP0266)

Redfish allows a POST request to a ResourceCollection to be equivalent to the same POST request to the Members property of that ResourceCollection. For a particular ResourceCollection, if a Swordfish implementation supports either form, it shall support both.

It is common practice in Redfish to also eliminate the Members property from any request URI that navigates through a type hierarchy that includes a Member within a ResourceCollection. Care should be taken when defining and using a ResourceCollection subclass to not introduce ambiguities when an explicit reference to a Members property is dropped from a request URI.

9 Swordfish type definitions

9.1 Overview

The following sections define the schema and type definitions that make up a Swordfish implementation. Each data type or entity within the schema includes a description that defines its implementation requirements and their interaction

9.2 Common properties

This section describes the properties (data fields) that share a common definition across many or all Redfish schema

9.2.1 Properties defined for all Redfish schemas

The following properties are included in every Redfish schema, and therefore may be encountered in any Response payload. They are documented here to avoid repetition in the Resource Guide tables for each schema.

Table 9: Common Properties

Property	Datatype	Attributes	Notes
@odata.context	string	read-only	The @odata.context property is a URL to a metadata document with a fragment describing the data (typically rooted at the top-level singleton or collection). Technically the metadata document only has to define, or reference, any of the types that it directly uses, and different payloads could reference different metadata documents. However, since the @odata.context provides a root URL for resolving relative references (such as @odata.id's), we return the canonical metadata document.
@odata.id	string	read-only	The @odata.id property is a string that indicates the unique identifier of a resource.
@odata.type	string	read-only	The @odata.type property is a URL fragment that indicates the type of the resource.
Description	string	read-write	The Description property is used to convey a human-readable description of the resource.
Id	string	read-write	The Id property of a resource uniquely identifies the resource within the Resource Collection that contains it. The value of Id is unique within a Resource Collection.
Name	string	read-write	The Name property is used to convey a human-readable moniker for a resource. The type of the Name property is a string. The value of Name is NOT necessarily unique across resource instances within a Resource Collection.

Property	Datatype	Attributes	Notes
Oem { }	object	read-write	This is the manufacturer/provider specific extension moniker used to divide the Oem object into sections. See the Resource schema for details on this property.

9.2.2 Links

The Links property represents the links associated with the resource, as defined by that resource's schema definition. All associated reference properties defined for a resource are nested under the Links property. All directly referenced (subordinate) properties defined for a resource can be found from the root of the resource.

9.2.3 Actions

The Actions property contains the actions supported by a resource.

9.2.4 OEM

The OEM property is used for OEM extensions.

9.2.5 RelatedItem

The RelatedItem property is represented as a set of links. The links point to a resource, or part of a resource, as defined by that resource's schema definition.

This representation is not intended to be a strong linking methodology like other references. Instead it is used to show a relationship between elements or sub-elements in disparate parts of the service. For example, Fans may be in one area of the system and Processors in another area of the system. It could be that the relationship between the two is not obvious. The RelatedItem property can be used to show that one is related to the other. In this example, it might indicate that a specific fan is cooling a specific processor.

9.2.6 Status

The `Status` property is common to many Redfish schema.

Health	string (enum)	read-only	This represents the health state of this resource in the absence of its dependent resources. See Health in Property Details, below, for the possible values of this property.
HealthRollup	string (enum)	read-only	This represents the overall health state from the view of this resource. See HealthRollup in Property Details, below, for the possible values of this property.

Oem { }	object	read-write	Oem extension object.
State	string (enum)	read-only	This indicates the known state of the resource, such as if it is enabled. See State in Property Details, below, for the possible values of this property.

9.2.6.1 Property details

Health:

This represents the health state of this resource in the absence of its dependent resources.

string	Description
Critical	A critical condition exists that requires immediate attention.
OK	Normal.
Warning	A condition exists that requires attention.

HealthRollup:

This represents the overall health state from the view of this resource.

string	Description
Critical	A critical condition exists that requires immediate attention.
OK	Normal.
Warning	A condition exists that requires attention.

State:

This indicates the known state of the resource, such as if it is enabled.

string	Description
Absent	This function or resource is not present or not detected.
Disabled	This function or resource has been disabled.
Enabled	This function or resource has been enabled.
InTest	This function or resource is undergoing testing.
Quiesced	The element is enabled but only processes a restricted set of commands.
StandbyOffline	This function or resource is enabled, but awaiting an external action to activate it.
StandbySpare	This function or resource is part of a redundancy set and is awaiting a failover or other external action to activate it.
Starting	This function or resource is starting.
UnavailableOffline	This function or resource is present but cannot be used.
Updating	The element is updating and may be unavailable or degraded.

9.2.7 Location

AltitudeMeters	number (m)	read-only (null)	The altitude of the resource in meters.
Info	string	read-only (null)	This indicates the location of the resource.
InfoFormat	string	read-only (null)	This represents the format of the Info property.
Latitude	number (deg)	read-only (null)	The latitude resource.
Longitude	number (deg)	read-only (null)	The longitude resource in degrees.
Oem { }	object	read-write	Oem extension object. See the Resource schema for details on this property.
PartLocation {	object	read-write	Postal address of the addressed resource.
LocationOrdinalValue	number	read-only (null)	The number that represents the location of the part. If LocationType is slot and this unit is in slot 2 then the LocationOrdinalValue will be 2.
LocationType	string (enum)	read-only	The type of location of the part, such as slot, bay, socket and slot. See LocationType in Property Details, below, for the possible values of this property.
Orientation	string (enum)	read-only	The orientation for the ordering of the slot enumeration used by the LocationOrdinalValue property. See Orientation in Property Details, below, for the possible values of this property.
Reference	string (enum)	read-only	The reference point for the part location. This is used to give guidance as to the general location of the part. See Reference in Property Details, below, for the possible values of this property.
ServiceLabel }	string	read-only (null)	This is the label of the part location, such as a silk screened name or a printed label.
Placement {	object	read-write	A place within the addressed location.

Rack	string	read-write (null)	Name of a rack location within a row.
RackOffset	number	read-write (null)	Vertical location of the item in terms of RackOffsetUnits.
RackOffsetUnits	string (enum)	read-write	The type of Rack Units in use. See RackOffsetUnits in Property Details, below, for the possible values of this property.
Row }	string	read-write (null)	Name of row.
PostalAddress {	object	read-write	Postal address of the addressed resource.
AdditionalCode	string	read-write (null)	Additional code.
Building	string	read-write (null)	Name of the building.
City	string	read-write (null)	City, township, or shi (JP).
Community	string	read-write (null)	Postal community name.
Country	string	read-write (null)	Country.
District	string	read-write (null)	A county, parish, gun (JP), or district (IN).
Division	string	read-write (null)	City division, borough, dity district, ward, chou (JP).
Floor	string	read-write (null)	Floor.
GPSCoords	string	read-write (null)	The GPS coordinates of the part.

HouseNumber	number	read-write (null)	Numeric portion of house number.
HouseNumberSuffix	string	read-write (null)	House number suffix.
Landmark	string	read-write (null)	Landmark.
LeadingStreetDirection	string	read-write (null)	A leading street direction.
Location	string	read-write (null)	Room designation or other additional info.
Name	string	read-write (null)	Name.
Neighborhood	string	read-write (null)	Neighborhood or block.
POBox	string	read-write (null)	Post office box (P.O. box).
PlaceType	string	read-write (null)	A description of the type of place that is addressed.
PostalCode	string	read-write (null)	Postal code (or zip code).
Road	string	read-write (null)	A primary road or street.
RoadBranch	string	read-write (null)	Road branch.
RoadPostModifier	string	read-write (null)	Road post-modifier.
RoadPreModifier	string	read-write (null)	Road pre-modifier.

RoadSection	string	read-write (null)	Road Section.
RoadSubBranch	string	read-write (null)	Road sub branch.
Room	string	read-write (null)	Name or number of the room.
Seat	string	read-write (null)	Seat (desk, cubicle, workstation).
Street	string	read-write (null)	Street name.
StreetSuffix	string	read-write (null)	Avenue, Platz, Street, Circle.
Territory	string	read-write (null)	A top-level subdivision within a country.
TrailingStreetSuffix	string	read-write (null)	A trailing street suffix.
Unit }	string	read-write (null)	Name or number of the unit (apartment, suite).

9.2.7.1 Property details

LocationType:

The type of location of the part, such as slot, bay, socket and slot.

string	Description
Bay	Defines a bay as the type of location.
Connector	Defines a connector as the type of location.
Slot	Defines a slot as the type of location.
Socket	Defines a socket as the type of location.

Orientation:

The orientation for the ordering of the slot enumeration used by the LocationOrdinalValue property.

string	Description
BackToFront	Defines the ordering for the LocationOrdinalValue is back to front.
BottomToTop	Defines the ordering for the LocationOrdinalValue is bottom to top.
FrontToBack	Defines the ordering for the LocationOrdinalValue is front to back.
LeftToRight	Defines the ordering for the LocationOrdinalValue is left to right.
RightToLeft	Defines the ordering for the LocationOrdinalValue is right to left.
TopToBottom	Defines the ordering for the LocationOrdinalValue is top to bottom.

RackOffsetUnits:

The type of Rack Units in use.

string	Description
EIA_310	Defines a rack unit as being equal to 1.75 in (44.45 mm).
OpenU	Defines a rack unit as being equal to 48 mm (1.89 in).

Reference:

The reference point for the part location. This is used to give guidance as to the general location of the part.

string	Description
Bottom	Defines the part as being in the bottom of the unit.
Front	Defines the part as being in the front of the unit.
Left	Defines the part as being in the left of the unit.
Middle	Defines the part as being in the middle of the unit.
Rear	Defines the part as being in the rear of the unit.
Right	Defines the part as being in the right of the unit.
Top	Defines the part as being in the top of the unit.

9.3 Complex Types

The following table defines a number of complex types that are used frequently in Swordfish schema. Multiple references to each complex type may be seen in later sections. For detailed definitions and properties contained in each complex type, refer to the schema definitions as referenced in the table.

Capacity {}	This composition may be used to represent storage capacity. The sum of the values in Data, Metadata,and Snapshot shall be equal to the total capacity for the datastore. See the Capacity.v1_1_0 schema for details.
CapacityInfo {}	This composition may be used to represent the utilization of storage capacity. See the Capacity.v1_1_0 schema for details.

IOStatistics {}	See the IOStatistics.v1_0_1 schema for details on this property.
IOWorkload {}	This structure may be used to describe an IO Workload. See the IOPerformanceLoSCapabilities.v1_0_0 schema for details.
IOWorkloadComponent{}	This structure may be used to describe a component of an IO workload. See the IOPerformanceLoSCapabilities.v1_1_1 schema for details.
ReplicaInfo {}	The value shall define the characteristics of a replica. See the StorageReplicaInfo.v1_1_0 schema for details.
ReplicaRequest {}	See the DataProtectionLineOfService.v1_1_0 schema for details.
Schedule {}	Schedule a series of occurrences. See the Schedule.v1_1_0 schema for details.