|  |  |
| --- | --- |
| **(Confidential)** | |
| Scope of disclosure | [TSDV] [TDSL] Storage PJ |
| Period of confidentiality | 7 years after release. |
| Head of Information Owner | Head of Engineering Dept. |
| Handling restriction | N/A |

**Remote Management**

**Investigation Report**

**SNIA Swordfish**

**Toshiba Software Development (Vietnam) Co., Ltd**

|  |
| --- |
| Document ID: TSDV-STORAGE-RM-Investigation Report |
| Total: 13 Page No. 1 |

**Revision History**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rev. No.  (X.YY) | Date (YYYY-MM-DD) | Section No. Changed | Change Description | Author | Reviewed by | Approved by |
| 0.01 | 2017-11-08 | All | Create new | TuNV |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Table of Contents**

[1. Introduction 6](#_Toc497900565)

[1.1. Outline 6](#_Toc497900566)

[1.2. Definitions and Acronyms 6](#_Toc497900567)

[2. Introduction to SNIA Swordfish 7](#_Toc497900568)

[2.1. About the SNIA 7](#_Toc497900569)

[3. Reference 8](#_Toc497900570)

**List of tables**

[Table 1: Definitions and Acronyms 6](#_Toc497900571)

**List of figures**

# Introduction

## Outline

This document is Swordfish investigation results to support TDSL customer define improvement direction for Remote Management. Because of effort limitation, the investigation focus on below topics:

* Chapter 2: Introduction to SNIA Swordfish
  + About SNIA
  + About Swordfish
  + Technology that Swordfish use
* Chapter 3: OData Protocol
  + OData Protocol
    - Overview
    - Data model and service model
    - Context URL
    - Security consideration
* Chapter 4: Redfish® API specification
  + Overview
  + Protocols detail
  + Data model and schema
  + Service detail
  + Security
* Chapter 5: Swordfish specification
  + Overview
  + Data model
  + Schema consideration
  + Implement requirement
  + Swordfish type definitions
* Chapter 6: Other vendors
  + Pure Storage
    - What is Pure and Pure1
    - About Pure1
  + Nimble Storage
    - What is Nimble and InfoSight
    - About InfoSight
* Chapter 7: Conclusion
  + Advantages and disadvantage of Swordfish API specification
  + What information will be needed for Remote Management
  + What technologies will be used in Remote Management
* Chapter 8: References

## Definitions and Acronyms

|  |  |  |
| --- | --- | --- |
| **No** | **Acronyms** | **Definition** |
| 1 | CST |  |
| 2 | SNIA |  |
| 3 | Swordfish  (SNIA Swordfish) |  |
| 4 | API |  |
| 5 | JSON |  |
| 6 | DMTF |  |
| 7 | JSON |  |
| 8 | OData |  |
| 9 | TWG | Technical working group |
|  | IT | Information technology |
|  |  |  |

Table 1: Definitions and Acronyms

(Refer: <https://swordfish> )

# Introduction to SNIA Swordfish

Target of this part is to talk about:

* About the SNIA (2.1)

## About the 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. [1]

SNIA has many technology communities and working groups, such as Cloud Storage Initiative, Green Storage Initiative, Green Storage TWG… Among them, the Scalable Storage Management (SSM) TWG is where the Swordfish API specification come from.

Scalable Storage Management TWG defined Swordfish API specification to help communities to standard the process of managing Scalable Storage by using a Web-based API and modern data centers.

## About Swordfish

SNIA Swordfish™ is a standard API, making it easier for IT administrators integrate scalable solutions (storage) into their data centers, and help IT engineer to easily implement their web-based storage management system.

SNIA Swordfish has been designed around management use cases that focus on what IT administrators need to do with storage equipment and storage services in a data center. As a result, the API provides functionality that simplifies the way storage can be allocated, monitored, and managed. [2]

Besides Swordfish, SNIA none-profit organization is working on many other technology related to information technology. Image below show the SNIA Organization and location of Scalable Storage Management TWG in it.

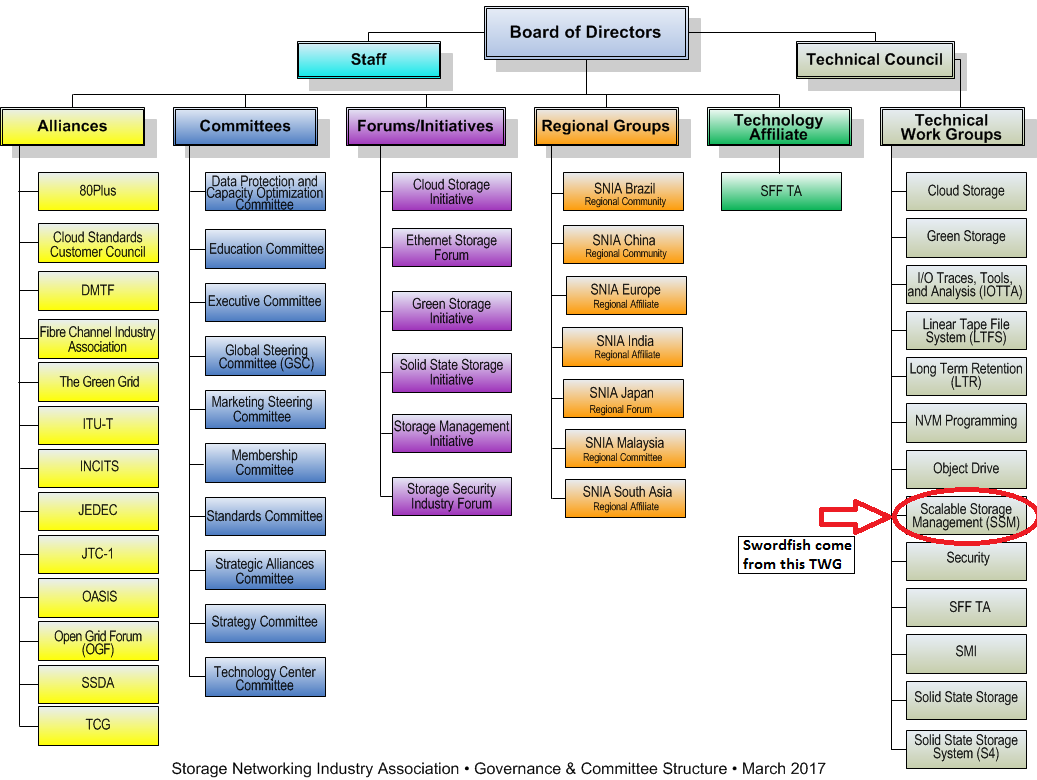


Figure 1: SNIA organization

(Refer: <https://www.snia.org/about/organization> )

## Technology that Swordfish use

SNIA Swordfish is designed to integrate with the technologies used in cloud data center environments and can be used to accomplish a broad range of storage management tasks from the simple to the advanced. [2]

SNIA Swordfish is an extension of the DMTF Redfish specification, so it use RESTful interface, JavaScript Object Notation (JSON), Open Data Protocol (OData). RESTful and JSON are familiar technologies, this document will provide some knowledge about Redfish specification and OData for easy to follow the other sections.

Some references to the technologies that Swordfish use:

<http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part1-protocol.html> : OData Part 1: Protocol

<http://redfish.dmtf.org/> : DMTF Redfish page.

# OData Protocol

## Overview

The OData Protocol is an application-level protocol for interacting with data via RESTful interfaces. The protocol help us to describe data models, querying and editing data with the easy-to-use RESTful technology. It provides facilities for:

* Metadata: a description of the data model.
* Data: data entities and the relationships between them.
* Querying: request to get data from data center with some utilities such as filter, search, count data…
* Editing: creating, updating, and deleting data.
* Operations: invoking custom logic, such as turn of a computer via web interface.
* Vocabularies: attaching custom semantics.

The OData Protocol is different from other REST-based web service approaches in that it provides a uniform way to describe both the data and the data model. This improves interoperability between systems and allows an ecosystem to emerge. [3]

## Context URL

Context URL is the URL that describes what data or action you want to request from the service center. OData provide context URL template that is an optimized formats of Restful API, helping systems that implement OData can easily be communicate and integrated. The context URL template uses the following terms:

* {context-url} is the resource path to the $metadata document, an OData metadata document is a representation of the data model that describes the data and operations exposed by an OData service,
* {entity-set} is the name of an entity set that client want to retrieve from the OData service center,
* {entity} is the URL for an entity, use to retrieve an entity from OData service,
* {singleton} is the URL for a singleton entity,
* {select-list} is an optional parenthesized comma-separated list of selected properties, functions and actions,
* {property-path} is the path to a structural property of the entity,
* {type-name} is a qualified type name,
* {/type-name} is an optional type-cast segment containing the qualified name of a derived or implemented type prefixed with a forward slash.

### Service root

Context URL template: {context-url}

According to OData, all REST APIs should have a single entry point from which a generic end point client can navigate to the resources in the service. After get the service root, in the response we will see links to various resources in the service (links to $metadata document – service document, links to retrieve other collections of object and to other items)

Example of service root URL: <http://192.168.119.4/SwordFish/StorageService/>

Response for this URL could be:



Figure 2: Service root in OData

In the response, you can see the link to the metadata document with the key @odata.context, and to see how to use this response to retrieve other resource, see the sections below.

### Service document

Context URL template: {context-url}

Service document (metadata document) is the document that describe the service model including type schemas, available operations, etc. The context URL of the service document is the metadata document URL of the service. For example: <http://192.168.119.4/SwordFish/StorageService/$metadata>. And the response can be:



Figure 3: Metadata document

### Collection of Entities

Context URL template: {context-url}#{entity-set}

Collection of entities is a set of entity that we want to retrieve from the service. From the service document we can construct the URL for collection of entity by append the name of the entity set to the service root URL, for example: [http://192.168.119.4/SwordFish/StorageService/People](http://192.168.119.4/SwordFish/StorageService/Peoples), sample response can be:



Figure 4: OData collections of entities

In this response, the @odata.context URL is the URL to retrieve the people metadata, the @odata.nextLink URL is the URL that is used to retrieve the next collection of People, because each request only return 8 People. Each @odata.id URL is used to retrieve the single People with the name in the bracket.

### Entity

Context URL template: {context-url}#{entity-set}/$entity

To get a particular entity from a collection, append a key segment. By default, key segments in OData services are bounded by parentheses because they may be composite keys, e.g., OrderLine(OrderId=1,LineNumber=1) or alternate keys, e.g., Person(SSN='000-00-0000') and Person(2115) both address the same resource. Some OData services use normal URL segments for key segments, e.g., Orders/1. This is not recommended because of the scenarios mentioned above. [4]

Example of Entity URL: <http://192.168.119.4/SwordFish/StorageService/People('russellwhyte')> where ‘russellwhyte’ is the name of the people we want to retrieve.

## Data service request

This section describes some information about HTTP method GET for OData resources. For full reference about POST, PATCH, PUT, and DELETE, see OData document at <http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part1-protocol.html>.

### System query option

OData defines a number of system query options that allow refining the request. System query options are prefixed with the dollar ($) character, which is optional in OData 4.01. 4.01 services MUST support case-insensitive system query option names specified with or without the $ prefix. [3]

Example of OData build-in query option: You want to get a set of People that the first name equal to Vincent. The URL to request that is: http://192.168.119.4/SwordFish/StorageService/People?$filter=FirstName eq 'Vincent’.

Other common query option can be:

* $select: requests that the service return only the properties, dynamic properties, actions and functions explicitly requested by the client. For example: <http://192.168.119.4/SwordFish/StorageService/People?$select=FirstName,LastName>
* $expand: use to select related entities and information. For example: [http://192.168.119.4/SwordFish/StorageService/People(‘javieralfred’)?$expand=Friends](http://192.168.119.4/SwordFish/StorageService/People('javieralfred')?$expand=Friends) will return information about the people named ‘javieralfred’ and his/her friend information.
* $orderby: specifies the order in which items are returned from the service. For example: http://192.168.119.4/SwordFish/StorageService/People?$orderby=length(FirstName) desc
* $top: the $top system query option specifies the number of entity that you want to retrieve from the service. For example: <http://192.168.119.4/SwordFish/StorageService/People?$top=5> will return fist 5 People Entity.
* $skip: specifies the number of entity that excludes the first n items of the queried collection from the result.
* $search: request only entity that satisfy the search criteria. For example: <http://192.168.119.4/SwordFish/StorageService/People?$search=John> OR Tim

For full query options, see OData document, section 11.2.1

### Built-in Filter Operations

In version 4.01, OData service must support case-insensitive (not case-sensitive in version 4.0) filter operation as describe below:

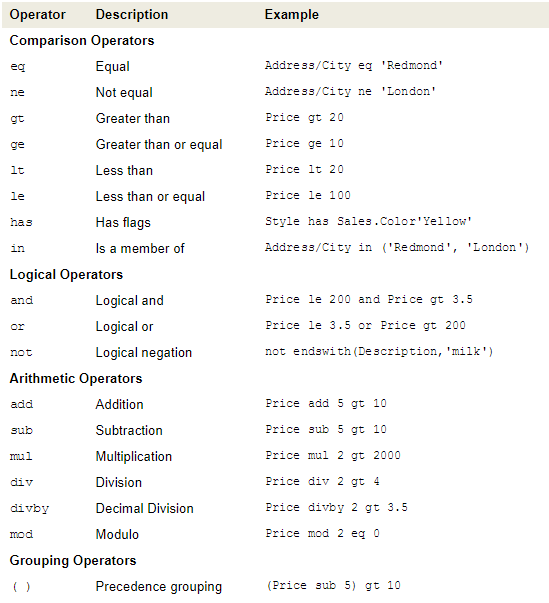


Figure 5: Filter operations

### Built-in Query Functions

OData supports a set of built-in functions that can be used within $filter operations. The following table lists the available functions.

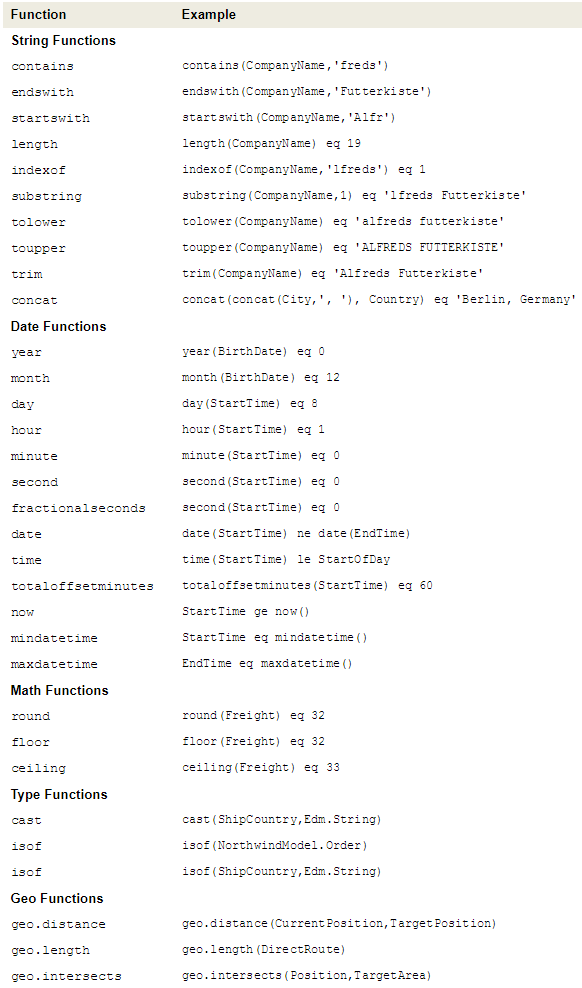


Figure 6: Query Functions

# Redfish API specification

## Overview

The Redfish Scalable Platforms Management API ("Redfish") is a new specification that uses RESTful interface semantics to access data defined in model format to perform out-of-band systems management. It is suitable for a wide range of servers, from stand-alone servers to rack mount and bladed environments but scales equally well for large scale cloud environments. [5]

Redfish is model oriented, and it is based on REST, so it is easier to use and implement, extend and maintain than many other solutions. By using human-readable JSON data format, it is intuitive for a human to examine the data. Redfish provide a standard for out-of-band systems management to easy communicate and merge.

Redfish architecture has many goals such as scalable (support many type of equipment in cloud service environments), flexible (support variety of systems type), standards based (to leverage protocols and standards that are widely accepted and used in environments), simple, lightweight, all aim to provide the community with the best standard of system management.

Redfish API and schemas follow OData conventions with the set of common RESTful conventions which provides for interoperability between APIs.

Redfish is model-oriented, resources are defined in OData Schema representation and translated to JSON Schema representation to optimize best practices for RESTful services.

## Protocols detail

The Redfish protocol is designed around a web service based interface model, and based on HTTP. HTTP methods are used by the Redfish protocol for create, read, update and delete operations. Actions and media types are extended part from HTTL which will be discuss later.

### HTTP methods

The following table describes the general mapping of operations to HTTP methods shall be supported by a Redfish interface:

|  |  |
| --- | --- |
| HTTP Method | Interface Semantic |
| POST | Object create, Object action, Eventing |
| GET | Object retrieval |
| PATCH | Object update |
| DELETE | Object delete |

Table 2: Redfish required HTTP method

As we can see, the PUT method is not required by Redfish, but system can provide it for Object replace action.

## Data model and schema

## Service detail

## Security

# Swordfish specification

## Overview

## Data model

## Schema consideration

## Implement requirement

## Swordfish type definitions

# Other vendors

## Pure Storage

### What is Pure and Pure1

### About Pure1

## Nimble Storage

### What is Nimble and InfoSight

### About InfoSight

# Conclusion

## Advantages and disadvantages of Swordfish API specification

## What information will be needed for Remote Management

## What technologies will be used in Remote Management

# References

|  |  |  |
| --- | --- | --- |
| **Index** | **Reference** | **Description** |
| 1 | <https://www.snia.org/about> | About SNIA website |
| 2 | <https://www.snia.org/forums/smi/swordfish> | SNIA Swordfish page |
| 3 | odata-v4.01-part1-protocol.pdf | OData Version 4.01. Part 1: Protocol |
| 4 | <http://www.odata.org/getting-started/learning-odata-on-postman/> | OData tutorial |
| 5 | DSP0266\_1.3.0.pdf | Redfish Scalable Platforms Management API Specification |