Swordfish Scalable Storage Management API
User’s Guide

Version 1.2.2

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.

Publication of this Working Draft for review and comment has been approved by the Scalable Storage Management Technical Work Group. This draft represents a ‘best effort’ attempt by the Scalable Storage Management Technical Work Group to reach preliminary consensus, and it may be updated, replaced, or made obsolete at any time. This document should not be used as reference material or cited as other than a ‘work in progress.’ Suggestions for revision should be directed to http://www.snia.org/feedback.

Working Draft

Last Updated 2 March 2021
USAGE

Copyright (c) 2016 - 2021 SNIA. All rights reserved. All other trademarks or registered trademarks are the property of their respective owners.

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

1. Any text, diagram, chart, table or definition reproduced must be reproduced in its entirety with no alteration, and,

2. Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced must acknowledge the SNIA copyright on that material, and must credit the SNIA for granting permission for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, or any portion thereof, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by emailing tcmd@snia.org. Please include the identity of the requesting individual and/or company and a brief description of the purpose, nature, and scope of the requested use.

All code fragments, scripts, data tables, and sample code in this SNIA document are made available under the following license:

BSD 3-Clause Software License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of The Storage Networking Industry Association (SNIA) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

Swordfish Scalable Storage Management API Users Guide  Working Draft  Version 1.2.2 (2 March 2021)
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.

DISCLAIMER

The information contained in this publication is subject to change without notice. The SNIA makes no warranty of any kind
with regard to this publication, including, but not limited to, the implied warranties of merchantability and fitness for a
particular purpose. The SNIA shall not be liable for errors contained herein or for incidental or consequential damages in
connection with the furnishing, performance, or use.

Suggestions for revisions should be directed to http://www.snia.org/feedback/.
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.
## Revision History

The evolution of this document is summarized in [Table 1](#).

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>19 September 2016</td>
<td>1.0.0</td>
<td>Initial Release</td>
</tr>
<tr>
<td>12 October 2016</td>
<td>1.0.1</td>
<td>General clean up and formatting consistency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A discussion of unused CoS and LoS entries in ServiceCatalog</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Improve purpose for many use cases</td>
</tr>
<tr>
<td>1 November 2016</td>
<td>1.0.2</td>
<td>Corrected XREF link formatting</td>
</tr>
<tr>
<td>24 January 2017</td>
<td>1.0.3</td>
<td>Additional use cases and new document section addressing client</td>
</tr>
<tr>
<td></td>
<td></td>
<td>considerations</td>
</tr>
<tr>
<td>25 April 2017</td>
<td>1.0.4</td>
<td>Update cross-references</td>
</tr>
<tr>
<td>3 October 2017</td>
<td>1.0.5</td>
<td>Minor updates and corrections</td>
</tr>
<tr>
<td>13 February 2018</td>
<td>1.0.6</td>
<td>Added on-demand replication use cases</td>
</tr>
<tr>
<td>12 October 2018</td>
<td>1.0.7</td>
<td>Editorial cleanup of JSON</td>
</tr>
<tr>
<td>22 August 2019</td>
<td>1.1.0</td>
<td>Restructured to add features and feature cross references, and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>many new use cases added:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create Volume for multiple scenarios (including Redfish Storage)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create Storage Pool for multiple scenarios</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replication use cases using single Volume</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replication use cases using Consistency Groups</td>
</tr>
<tr>
<td>12 November 2019</td>
<td>1.1.0</td>
<td>Released as Technical Position</td>
</tr>
<tr>
<td>12 November 2019</td>
<td>1.1.0a</td>
<td>Released as Corrected Technical Position</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Formatting fixes – word wrap in pdf doc format to fix truncated lines</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added cross referencing of Features to use cases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Editorial changes and cleanup</td>
</tr>
<tr>
<td>18 August 2020</td>
<td>1.2.1</td>
<td>Added NVMe-specific use cases</td>
</tr>
<tr>
<td>31 October 2020</td>
<td>1.2.1c</td>
<td>Released as SNIA Approved Publication</td>
</tr>
<tr>
<td>2 March 2021</td>
<td>1.2.2</td>
<td>Added cross-references to NVMe mapping document</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added new use cases for StoragePool actions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errata fixes.</td>
</tr>
</tbody>
</table>
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 listed in Table 2.

Table 2: Contributors

<table>
<thead>
<tr>
<th>Member</th>
<th>Representatives (* – prior employer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcom Inc.</td>
<td>Richelle Ahlvers*</td>
</tr>
<tr>
<td>Cisco Systems, Inc.</td>
<td>Krishnakumar Gowravaram</td>
</tr>
<tr>
<td>Dell Inc.</td>
<td>Patrick Boyd</td>
</tr>
<tr>
<td></td>
<td>George Ericson</td>
</tr>
<tr>
<td></td>
<td>Jim Pendergraft</td>
</tr>
<tr>
<td></td>
<td>Sean McGinnis</td>
</tr>
<tr>
<td></td>
<td>Michael Raineri</td>
</tr>
<tr>
<td></td>
<td>Rich Roscoe</td>
</tr>
<tr>
<td>Futurewei Inc.</td>
<td>Sean McGinnis*</td>
</tr>
<tr>
<td>Hitachi Data Systems</td>
<td>Eric Hibbard</td>
</tr>
<tr>
<td>Hewlett Packard Enterprise</td>
<td>Curtis Ballard</td>
</tr>
<tr>
<td></td>
<td>Jeff Hilland</td>
</tr>
<tr>
<td></td>
<td>Chris Lionetti</td>
</tr>
<tr>
<td></td>
<td>John Mendonca</td>
</tr>
<tr>
<td></td>
<td>Doug Voigt</td>
</tr>
<tr>
<td>Inova Development Inc.</td>
<td>Karl Schopmeyer</td>
</tr>
<tr>
<td>Member</td>
<td>Representatives</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>Intel Corporation</td>
<td>Richelle Ahlvers&lt;br&gt;Rajalaxmi Angadi&lt;br&gt;Klaudia Jablonska&lt;br&gt;Mariusz Krzywienski&lt;br&gt;Mateusz Mania&lt;br&gt;Slawek Putyrski&lt;br&gt;Paul von Behren</td>
</tr>
<tr>
<td>Microsemi Corporation</td>
<td>Anand Nagarjan</td>
</tr>
<tr>
<td>Microsoft Corporation</td>
<td>Hector Linares&lt;br&gt;Jim Pinkerton&lt;br&gt;Michael Pizzo&lt;br&gt;Scott Seligman</td>
</tr>
<tr>
<td>NetApp, Inc.</td>
<td>Don Deel&lt;br&gt;Fred Knight&lt;br&gt;Nilesh Maheshwari</td>
</tr>
<tr>
<td>ScienceLogic</td>
<td>Patrick Strick</td>
</tr>
<tr>
<td>VMware, Inc.</td>
<td>Murali Rajagopal</td>
</tr>
</tbody>
</table>
# Table of Contents

**USAGE**
- DISCLAIMER .......................... 2
- Contact SNIA .......................... 3

**FEEDBACK AND INTERPRETATIONS**

**INTENDED AUDIENCE**

**VERSIONING POLICY**

**Revision History**

**About SNIA**

**Acknowledgements**

**Table of Contents**

**1 Introduction**
- 1.1 Audience .......................... 9
- 1.2 Documentation structure ........ 9
- 1.3 Implementation scope assumptions 9
- 1.4 Base implementation assumptions 10
- 1.5 Knowledge assumptions .......... 10
- 1.6 Related documents ............... 11

**2 General query syntax**
- 2.1 Query method ...................... 12
- 2.2 Query Headers .................... 12
- 2.3 Service root ....................... 14
- 2.4 Resource path ..................... 14
- 2.5 Query options ..................... 15
- 2.6 Filter expressions ............... 15
- 2.7 HTTP status codes ............... 15

**3 Actors**
- 3.1 Actors ............................ 16
- 3.2 CloudAdmin ....................... 16
- 3.3 DevOps ............................ 16
- 3.4 Storage Admin .................... 17

**4 Management Domains**
- 4.1 Management Domain Overview .... 19
- 4.2 Application storage management domain 19
- 4.3 Block storage management domain 19
- 4.4 Service catalog management domain 20
- 4.5 File system storage management domain 22

**5 User Guidance**

**6 Features**
- 6.1 Overview .......................... 24
- 6.2 Block provisioning feature ...... 25
- 6.3 Capacity management feature ... 25
- 6.4 Class of Service Features ...... 25
- 6.5 Event notification feature ...... 26
- 6.6 File provisioning feature ...... 27
- 6.7 Block IO performance feature ... 27
- 6.8 Block mapping and masking feature 27
- 6.9 NVMe Support feature .......... 28
- 6.10 Replication Feature ............ 29

**7 Use cases**
- 7.1 Use cases by feature ............ 30
- 7.2 Alphabetic list of use cases .... 34
1 Introduction

1.1 Audience

This guide is intended to provide a common repository of best practices, common tasks and education for the users of the Swordfish API. Each use case includes an indication of the classes of API users who are most likely to find the case useful.

1.2 Documentation structure

This document begins with a set of information intended to provide a solid foundation for readers new to restful APIs in general and Swordfish in particular. While this material is no replacement for a thorough understanding of the Swordfish specification and the material that it references, it is intended as a stand alone document that can provide a solid introduction to Swordfish.

Based on that foundational material, this document then presents a set of Use Cases intended to capture common tasks and best practices that can be used to exercise the breadth and strength of the Swordfish API. In general, the guide is structured to provide more basic use cases first, and examine common refinements and options at the same time. More advanced tasks are handled later in the guide, and assume the prior skills and assumptions have been mastered.

For each use case, this guide will use a common template. Table 3 lists each field of the template and its description.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>A description of the high-level scope of the Use Case</td>
</tr>
<tr>
<td>Summary</td>
<td>A high-level summary of the use case</td>
</tr>
<tr>
<td>Purpose</td>
<td>The intended goals or motivations for the use case</td>
</tr>
<tr>
<td>Who</td>
<td>The Actor(s) who are likely to use this use case. The actor description also includes a list of other use cases of interest for a named actor</td>
</tr>
<tr>
<td>Management Domain</td>
<td>The Management Domain(s) applicable to this use case. The domain description also includes a list of other use cases of interest for the named management domains</td>
</tr>
<tr>
<td>Triggers</td>
<td>A description of likely business conditions or goals that would make this use case useful</td>
</tr>
<tr>
<td>Detailed Context</td>
<td>A detailed description of the operations environment and configuration assumptions for this use case</td>
</tr>
<tr>
<td>Preconditions</td>
<td>Pre-existing knowledge, configurations or capabilities</td>
</tr>
<tr>
<td>Inputs</td>
<td>A set of parameters and values that are used to adapt a generic use case to a specific business needs. Where appropriate, the parameter description will include a data type (e.g., (CAPACITY): desired storage capacity (int64))</td>
</tr>
<tr>
<td>Basic Course of Events</td>
<td>A sequence of API requests, including required headers, the body of the request, and the expected reply</td>
</tr>
<tr>
<td>Configuration Impacts</td>
<td>Changes to the storage configurations caused by the use case</td>
</tr>
<tr>
<td>Failure Scenario</td>
<td>Common failure conditions encountered in this use case</td>
</tr>
<tr>
<td>See Also</td>
<td>Other Use Cases that may be of interest</td>
</tr>
</tbody>
</table>

### 1.3 Implementation scope assumptions

The precise scope of a given Swordfish implementation can vary widely. Some installations will opt to deploy a basic level of the API that only extends the Redfish standard slightly. Others will decide to implement a number of features, providing a broader range of functionality. While this guide cannot provide examples of all possible configurations and situations, it does attempt to cover a range of possible functionality options. Use cases that assume functionality that correspond to specific features are clearly identified.

### 1.4 Base implementation assumptions

This document assumes that some fundamental configuration issues have been properly implemented, and will not need to be addressed in any detail. In particular, this document assumes:

- An appropriate security infrastructure (e.g., TLS 1.2)
- A functional Swordfish/Redfish installation, in either a standalone, aggregator, or distributed configuration
- Any required login credentials

### 1.5 Knowledge assumptions

The Swordfish API conforms to the standards defined in the Redfish API (http://redfish.dmtf.org). More generally, it provides a RESTful interface. The reader is assumed to be familiar with common conventions for RESTful APIs. Those readers who are interested in additional background information are encouraged to refer to the following sources:

- For Redfish standards: Redfish Specification (http://redfish.dmtf.org)
1.6 Related documents

This User's Guide is part of the documentation suite for the Swordfish API. Readers are encouraged to refer to the following for additional information:

- Swordfish API Specification (http://www.snia.org/swordfish)
- Swordfish Tutorials (http://www.snia.org/swordfish)
- Swordfish NMVe Mapping Guide (http://www.snia.org/swordfish)
- Swordfish Profile Bundle (http://www.snia.org/swordfish)
2 General query syntax

2.1 Query method

Swordfish queries support four query methods. Each query URL must include exactly one of the query methods listed in Table 4.

<table>
<thead>
<tr>
<th>Method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Retrieve the current state or settings of the named Resource Path as seen through the Service Root</td>
</tr>
<tr>
<td>POST</td>
<td>Create a new object under the named Resource Path</td>
</tr>
<tr>
<td>PUT</td>
<td>Replace the object referenced by the named Resource Path</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete the object referenced by the named Resource Path</td>
</tr>
<tr>
<td>PATCH</td>
<td>Update the object referenced by the named Resource Path</td>
</tr>
<tr>
<td>HEAD</td>
<td>Validates a GET request against the named Resource Path without returning the HTML headers for the response without the result of the query</td>
</tr>
</tbody>
</table>

2.2 Query Headers

All HTTP requests and responses in a compliant Swordfish implementation support the HTTP headers required by the Redfish Protocol Specification (http://redfish.dmtf.org/schemas/DSP0266_1.0.html). The supported headers are reproduced here for convenience.

2.2.1 Request headers

HTTP request headers that are commonly used in Swordfish queries are summarized in Table 5.

<table>
<thead>
<tr>
<th>Header</th>
<th>Supported Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>RFC 7231</td>
<td>Indicates to the server what media type(s) this client is prepared to accept. Services shall support requests for resources with an Accept header including application/json or application/json;charset=utf-8. Services shall support requests for metadata with an Accept header including application/xml or application/xml;charset=utf-8.</td>
</tr>
<tr>
<td></td>
<td>(<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231</a>)</td>
<td></td>
</tr>
<tr>
<td>Content-Type</td>
<td>RFC 7231</td>
<td>Describes the type of representation used in the message body. Content-Type shall be required in requests that include a request body. Services shall accept Content-Type values of application/json or application/json;charset=utf-8.</td>
</tr>
<tr>
<td></td>
<td>(<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231</a>)</td>
<td></td>
</tr>
</tbody>
</table>
Table 5: Request headers, cont.

<table>
<thead>
<tr>
<th>Header</th>
<th>Supported Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OData-Version</td>
<td>4.0</td>
<td>Services shall reject requests which specify an unsupported OData version. If a service encounters a version that it does not support, the service should reject the request with status code 412. If client does not specify an Odata-Version header, the client is outside the boundaries of this specification.</td>
</tr>
<tr>
<td>Authorization</td>
<td>RFC 7235 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7235">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7235</a>). Section 4.2</td>
<td>Required for Basic Authentication</td>
</tr>
<tr>
<td>User-Agent</td>
<td>RFC 7231 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231</a>)</td>
<td>Required for tracing product tokens and their version. Multiple product tokens may be listed.</td>
</tr>
<tr>
<td>Host</td>
<td>RFC 7230 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7230">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7230</a>)</td>
<td>Required to allow support of multiple origin hosts at a single IP address.</td>
</tr>
<tr>
<td>Origin</td>
<td>W3C CORS, Section 5.7 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#W3C-CORS">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#W3C-CORS</a>)</td>
<td>Used to allow web applications to consume Redfish Service while preventing CSRF attacks.</td>
</tr>
<tr>
<td>If-Match</td>
<td>RFC 7232 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7232">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7232</a>)</td>
<td>If-Match shall be supported on PUT and PATCH requests for resources for which the service returns ETags, to ensure clients are updating the resource from a known state.</td>
</tr>
<tr>
<td>X-Auth-Token</td>
<td>Opaque encoded octet strings</td>
<td>Used for authentication of user sessions. The token value shall be indistinguishable from random.</td>
</tr>
</tbody>
</table>

2.2.2 Response headers

HTTP response headers that are commonly used in Swordfish queries are summarized in Table 6.

Table 6: Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Supported Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OData-Version</td>
<td>4.0</td>
<td>Describes the OData version of the payload that the response conforms to.</td>
</tr>
<tr>
<td>Content-Type</td>
<td>RFC 7231 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231</a>)</td>
<td>Describes the type of representation used in the message body. Services shall specify a Content-Type of application/json when returning resources as JSON, and application/xml when returning metadata as XML. ;charset=utf-8 shall be appended to the Content-Type if specified in the chosen media-type in the Accept header for the request.</td>
</tr>
<tr>
<td>ETag</td>
<td>RFC 7232 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7232">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7232</a>)</td>
<td>An identifier for a specific version of a resource, often a message digest. Etags shall be included on responses to GETs of ManagerAccount objects.</td>
</tr>
<tr>
<td>Server</td>
<td>RFC 7231 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231</a>)</td>
<td>Required to describe a product token and its version. Multiple product tokens may be listed.</td>
</tr>
<tr>
<td>Link</td>
<td></td>
<td>Link headers shall be returned as described in the clause on LinkHeaders (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#link-header">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#link-header</a>)</td>
</tr>
<tr>
<td>Header</td>
<td>Supported Values</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Location</td>
<td>RFC 7231 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7231</a>)</td>
<td>Indicates a URI that can be used to request a representation of the resource. Shall be returned if a new resource was created. Location and X-Auth-Token shall be included on responses which create user sessions.</td>
</tr>
</tbody>
</table>

Table 6: Response headers, cont.

<table>
<thead>
<tr>
<th>Header</th>
<th>Supported Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache-Control</td>
<td>RFC 7234 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7234">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7234</a>)</td>
<td>This header shall be supported and is meant to indicate whether a response can be cached or not.</td>
</tr>
<tr>
<td>Access-Control-Allow-Origin</td>
<td>W3C CORS, Section 5.7 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#W3C-CORS">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#W3C-CORS</a>)</td>
<td>Prevents or allows requests based on originating domain. Used to prevent CSRF attacks.</td>
</tr>
<tr>
<td>Allow</td>
<td>POST, PUT, PATCH, DELETE, GET, HEAD</td>
<td>Shall be returned with a 405 (Method Not Allowed) response to indicate the valid methods for the specified Request URI. Should be returned with any GET or HEAD operation to indicate the other allowable operations for this resource.</td>
</tr>
<tr>
<td>WWW-Authenticate</td>
<td>RFC 7234 (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7234">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#RFC7234</a>), Section 4.1</td>
<td>Required for Basic and other optional authentication mechanisms. See the Security (<a href="http://redfish.dmtf.org/schemas/DSP0266_1.0.html#security">http://redfish.dmtf.org/schemas/DSP0266_1.0.html#security</a>) clause for details.</td>
</tr>
<tr>
<td>X-Auth-Token</td>
<td>Opaque encoded octet strings</td>
<td>Used for authentication of user sessions. The token value shall be indistinguishable from random.</td>
</tr>
</tbody>
</table>

2.3 Service root

This is the base of all Swordfish URL’s. A GET request to the Service Root will return an overview of the services provided by a given Swordfish service. In addition, the Service Root will include versioning information.

All Service Root URLs that are compliant with the Swordfish specification will be of the form https://hostname/redfish/v1, where hostname specifies the system (and optionally port number), of the Swordfish service provider.

2.4 Resource path

The Resource Path identifies the specific object (or collection of objects) that is the target of the Swordfish query. Swordfish Resource Paths can identify:

- A singleton object (e.g., a specific storage LUN or Volume)
- A collection of objects (e.g., the list of all LUNs provided by a specific storage array)

At the highest level, Swordfish systems are discoverable in the Storage Systems collection in the ServiceRoot.
2.5 Query options

Swordfish queries can include arbitrary sets of query options to further refine the target of given query or the actions being requested of that target. These general query options are summarized in Table 7.

Note: Additional query options may be supported (or constrained) for a specific query target or resource path. These target-specific query options will be addressed in specific use case descriptions, as required.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Arguments</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$skip=n</td>
<td>Integer</td>
<td>Omit the first ( n ) entries in the collection from the returned set of objects (required by redfish (redfish.dmtf.org))</td>
</tr>
<tr>
<td>$top=n</td>
<td>Integer</td>
<td>Return, at most, the first ( n ) entries in the returned set of objects (required by redfish (redfish.dmtf.org))</td>
</tr>
<tr>
<td>$filter=condition</td>
<td>Filter Expression</td>
<td>Returns only the members of the named collection that match the provided logical expression (required by swordfish (snia.org/swordfish))</td>
</tr>
<tr>
<td>$expand=target</td>
<td>Expand Expression</td>
<td>Expand additional detail on the target property(s) in the returned result set (required by swordfish (snia.org/swordfish))</td>
</tr>
<tr>
<td>$select=property list</td>
<td>Comma-separated list of object properties</td>
<td>Return the named properties for each object in the result set, rather than the entire object (required by swordfish (snia.org/swordfish))</td>
</tr>
<tr>
<td>$orderby=filter condition</td>
<td>Filter Expression</td>
<td>sort the result set by the output values from the filter expression (required by swordfish (snia.org/swordfish))</td>
</tr>
</tbody>
</table>

2.6 Filter expressions

1. Simple example $filter=(age gt 30) A group of people never to trust.

For more information see Filter Expression (http://docs.oasis-open.org/odata/odata/v4.0/errata03/os/complete/part2-url-conventions/odata-v4-0-errata03-os-part2-url-conventions-complete.html#Toc453752356) in the OData specification.

2.7 HTTP status codes

Swordfish clients may receive any of the standard HTTP status codes. Both the Redfish specification and the Swordfish specification define a detailed mapping from the generic HTTP codes to domain-specific situations, and probable causes. In addition, the server can return extended status information as a simple JSON object to further clarify the handling and outcome of a particular API request. For more information, see the Swordfish Specification (http://snia.org/swordfish) and the Redfish Specification (http://redfish.dmtf.org).
3 Actors

3.1 Actors

This document covers a broad range of common use cases and storage management operations.

In an attempt to serve as both a introductory text and as a reference tool, the use cases have been grouped in a number of different ways:

- Alphabetically. Each use case is ordered according to a self-explanatory title, to make it easier for an experienced user to find a specific piece of guidance.
- By management domain. While each organization will allocate responsibilities differently, there are general broad classes of storage management that share tasks. These are called management domains.
- By actor. Similarly, the precise titles and responsibilities within storage management organizations will differ from one organization to the next, but this guide has identified three broad functional roles, and identified the use cases that are most applicable to those roles:
  - Cloud Admin: tasked with managing all aspects of cloud-based infrastructure including storage;
  - Storage Admin: tasked with operational storage tasks, such as storage lifecycle planning and day-to-day storage administration;
  - DevOps: tasked with leveraging storage configurations to automate and scale business operations.

3.2 CloudAdmin

A Cloud Administrator (CloudAdmin) is a converged infrastructure administrator, working with systems that are:

- Hyper-converged
- Rack-converged
- Hyper-scale

The CloudAdmin role in an enterprise or service provider is the individual or group primarily responsible for managing the operational lifecycle of a cloud, virtualization, converged environment that consists of the workloads, resource abstractions, storage, networking, and compute.

Also referred to as a “cloud architect”, this role:

- designates a group of people to build and maintain a virtualized, converged, and/or cloud environment
- spans compute, storage, and networking disciplines as their job is to manage the operational lifecycle of the entire environment
- focuses on automation that involves scripting and potentially formal programming
- This role’s value is to keep the environment that hosts applications available and responding. Deep subject matter expertise is less valuable unless it helps solve an issue.
- The focus for this role is to streamline the deployment of applications into the infrastructure including all the network and storage configuration.
- This role gets involved in the physical deployment of capacity in the datacenter.
- This role deals with software defined infrastructure deployment and management.
- Applications can span physical, virtual machine, container, PaaS building blocks. This role is expected to know how to best configure the “stack” to consume the underlying physical infrastructure.

3.2.1 Use Cases

- Create StorageGroup
- Create Volume from an Existing Storage Pool
- Create an on-demand snapshot of a Volume
- Create file share
- Subscribe to threshold events

3.3 DevOps

A member of the DevOps group:
Consumes infrastructure capacity offered as a service/building blocks.
Develops and deploys programmatic requests for capacity (fully automated, no ticketing or human intervention)
Provisions storage as a virtual appliance on-premises or in the cloud as part of a larger application deployment
Deploys ‘cloud born apps’ to consume object storage APIs (S3, Swift)

This role is typically aligned with the business unit. Their focus is the delivery, deployment, and maintenance of apps on IaaS and PaaS resources. This role is not typically a deep subject matter expert in compute, storage, or networking. From a development perspective, their desire is to treat the infrastructure as a programmable subsystem that presents resources on-demand.

This role:
- does not get involved in the physical deployment of capacity in the datacenter
- is responsible for automating infrastructure provisioning as part of the E2E deployment and management of an application with minimal or no human intervention
- consumes higher level services and protocols from the infrastructure; understanding how the device works is not interesting
- expects to consume the programmable interfaces of a device using their existing tools (REST, Python, Ansible, Puppet, Chef, Ruby etc.)

### 3.3.1 Use Cases

- Create a New Replication Relationship by Assigning a Target Volume
- Create a New Replication Relationship by Assigning a Target Consistency Group
- Create an on-demand snapshot of a Volume
- Create class of service
- Create file share
- Create file system
- Create line of service
- Create storage pool
- Create storage pool using Specified Set of Drives
- Create storage pool using Specified Set of Drives and RAIDTypes
- Create Volume from an Existing Storage Pool
- Expand capacity of a storage volume
- Health updates
- Make a New Replication Relationship by Creating a Target Consistency Group
- Make a New Replication Relationship by Creating a Target Volume
- Remove Replication Relationship
- Remove Replication Relationship for a Consistency Group
- Resume the Replication Synchronization Activity
- Resume the Replication Synchronization Activity for a Consistency Group
- Reverse a Replication Relationship
- Reverse a Replication Relationship for Consistency Groups
- Split a Replica
- Split a set of Replicas in Consistency Groups
- Subscribe to Threshold Events
- Suspend Replication Synchronization Activity
- Suspend Replication Synchronization Activity between Consistency Groups

### 3.4 Storage Admin

A storage administrator designs storage solutions for modern environments, including:
- Virtualization (traditional virtualization VMware, Hyper-V)
- private cloud (self-service portal)
- hybrid cloud (can span private, colo, hosted, and public cloud)
- IaaS/PaaS stacks (modern app/devops)
The storage admin role in an enterprise or service provider can have responsibility for managing the operational lifecycle of storage in the datacenter. In particular:

- Organizations can have one or more people exclusively assigned to operating lifecycle of storage in the datacenter.
- The storage admin role is typically tasked with “figuring out” storage needs for the company.
- The admin role can consist of level 2 operators and level 3 engineers and architects.
- The admin role deals with multiple storage devices from multiple vendors across one or more datacenters.
- Storage devices attached to multiple operating systems (Linux, Windows, Mainframe, Unix, virtualized, bare metal).
- The Admin role consists of deep storage subject matter expertise, may or may have general practitioners across all disciplines.
- Members of this role are typically skilled at scripting and formal programming.
- In a converged environment (storage, compute, and networking converged in to a rack), the storage admin role may choose to avoid/minimize involvement in the support of storage WITHIN the rack. (e.g. in virtualization, a single physical port can host 1000s of VMs).

### 3.4.1 Use Cases

- Create a New Replication Relationship by Assigning a Target Volume
- Create a New Replication Relationship by Assigning an existing Target Consistency Group
- Create an on-demand snapshot of a Volume
- Create class of service
- Create file share
- Create file system
- Create line of service
- Create storage pool
- Create storage pool using Specified Set of Drives
- Create storage pool using Specified Set of Drives and RAIDTypes
- Create Volume from an Existing Storage Pool
- Expand capacity of a storage volume
- Health updates
- Make a New Replication Relationship by Creating a Target Consistency Group
- Make a New Replication Relationship by Creating a Target Volume
- Remove Replication Relationship
- Remove Replication Relationship for a Consistency Group
- Resume the Replication Synchronization Activity
- Resume the Replication Synchronization Activity for a Consistency Group
- Reverse a Replication Relationship
- Reverse a Replication Relationship for Consistency Groups
- Split a Replica
- Split a set of Replicas in Consistency Groups
- Subscribe to Threshold Events
- Suspend Replication Synchronization Activity
- Suspend Replication Synchronization Activity between Consistency Groups
4 Management Domains

4.1 Management Domain Overview

This document covers a broad range of common use cases and storage management operations.

In an attempt to serve as both an introductory text and as a reference tool, the use cases have been grouped in a number of different ways:

- Alphabetically. Each use case is ordered according to a self-explanatory title, to make it easier for an experienced user to find a specific piece of guidance.
- By actor. While the precise titles and responsibilities within storage management organizations will differ from one organization to the next, this guide has identified three broad functional roles, known as *actors* and identified the use cases that are most applicable to those roles;
- By management domain. Similarly, while each organization will allocate responsibilities differently, there are general broad classes of storage management that share tasks. This guide has identified four management domains:
  - Application Storage Management: which manages the interface between applications and the storage that they rely upon;
  - Block Storage Management: focused on the management of resources that provide block-based access to storage;
  - Service Catalog Management: which supports access to, and management of, a catalog of service options.

4.2 Application storage management domain

This domain manages the interface between applications and the storage that they rely upon.

StorageGroups provide a means to collectively manage the Volumes and FileShares utilized by an Application. The StorageGroup specifies whether the collected resources are managed so that storage is updated or replicated consistently across all members. Additionally, the StorageGroup provides the means to atomically expose (or hide) the collected resources to (or from) host endpoints.

4.2.1 Use cases

- Subscribe to Threshold Events

4.3 Block storage management domain

Many devices and services provide their storage capacity to external devices and applications through block-based protocols to storage devices. This domain includes the management of resources that provide block-based access to storage.

Block-based storage is represented by a Volume. This domain provides for the discovery and provisioning of Volumes and for maintaining relationships to Device, Endpoint, StorageService, StorageGroup, StoragePool, and ComputerSystem resources.

4.3.1 Use Cases

- Create a New Replication Relationship by Assigning a Target Volume
- Create a New Replication Relationship by Assigning an existing Target Consistency Group
- Create an on-demand snapshot of a Volume
- Create class of service
- Create line of service
- Create storage pool
- Create storage pool using Specified Set of Drives
- Create storage pool using Specified Set of Drives and RAIDTypes
- Create StorageGroup
4.4 Service catalog management domain

Swordfish supports access to, and management of, a catalog of service options, (see ITIL glossary and abbreviations (https://www.axelos.com/Corporate/media/Files/Glossaries/ITIL_2011_Glossary_GB-v1-0.pdf)), supported by storage services.

When the Swordfish Class of Service Feature is implemented, the ClassOfService resource represents a service option that may be used to specify requirements for utility or warranty when provisioning a resource. Currently ClassOfService is defined for use in the Block Storage, File System, and ApplicationStorage domains.

The service catalog for each StorageService is represented by a collection of references to supported ClassOfService resources. Each ClassOfService is known minimally by a Name and a unique Identifier. When a ClassOfService is specified for a resource, the StorageService shall attempt to maintain that resource in compliance to the requirements of that ClassOfService. The requirements may be specified informally by text in the Description property or may be specified formally by the property values of embedded options related to specific lines of service.

The embedded service options are described by values of complex types representing lines of service.

Over time, as the service catalog is continually updated to match evolving user needs and service provider offerings, it is expected that the catalog will contain entries (one or more ClassOfService or LineOfService instances) that are not currently active.

4.4.1 Data protection

The primary storage is described by a ClassOfService resource. That ClassOfService may aggregate any number of data protection service options. Each instance of a data protection service option describes the characteristics a replication session that shall be maintained for the containing primary storage resource.

For additional information, see the definitions for DataProtectionLineOfService and DataProtectionLoSCapabilities.

4.4.2 Data security

An instance of a data security service option describes an optional set of data security requirements. A data security Service option is typically aggregated into a ClassOfService resource that is associated with storage. At most one data security service option may be aggregated into a ClassOfService resource. When storage is provisioned with that ClassOfService, it will provide the currently specified data security characteristics.

A data security service option may specify data security characteristics that 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. In this case, the names of the standard or standards can usefully be included in the user/admin-visible name of the instance. With the notable exception of FIPS-140, compliance
requires measures well beyond the means of a storage system to provide (e.g., both HIPAA and PCI impose significant requirements on administration and operation of the data center), so this approach promises that the storage system will do its part in supporting compliance, but does not (and cannot) promise that the storage system will deliver full compliance by itself.

The description attribute value may include human readable information including:

- Whether encryption keys are drive or array resident or externally managed (e.g., via KMIP).
- Information on how the array supports compliance to a standard identified in the name of the Service option. (e.g., specific algorithms employed that are FIPS-140 compliant, information about the validated cryptographic module and its validation certificate, relationship of the security functionality to specific PCI or HIPAA requirements).

NOTE For comparable cryptographic strengths, (see NIST SP 800-57 part 1 (http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf))

NOTE For symmetric encryption algorithm key sizes, 112 bits is the 3DES key size and 128, 192, and 256 bits are options for AES key sizes.

NOTE Media Encryption Strength includes the case where metadata about the data must be encrypted (e.g., data presence vs. absence in a thin volume, array filesystem metadata). The implementation may be self-encrypting drives or encryption in the storage system’s drive controller. Keys may be drive or array resident or externally managed (e.g., via KMIP).

For additional information, see the definitions for DataSecurityLineOfService and DataSecurityLoSCapabilities.

### 4.4.3 Data storage

Each data storage service option describes characteristics of the storage at a particular location. A class of service will have at most one data storage service option, which describes the storage specified by that class of Service.

For additional information, see the DataStorageLineOfService and DataProtectionLoSCapabilities.

### 4.4.4 IO connectivity

An IO connectivity service option specifies the characteristics of storage connectivity. For each value of AccessProtocol, at most one IO connectivity service option may be aggregated into a class of service.

NOTE: If used within a ClassOfService for Storage Provisioning, this value constrains the set of connections used to expose that storage.

For additional information, see the IOConnectivityLineOfService and IOConnectivityLoSCapabilities.

### 4.4.5 IO performance

An IO performance service option specifies a choice of performance characteristics as viewed through the data path to the storage. This is affected by choices of storage and connection technologies.

At most one IO performance service option may be aggregated into a ClassOfService for a storage resource. When storage is provisioned with that ClassOfService, it should provide at least the specified performance.

For additional information, see the IOConnectivityLineOfService and IOConnectivityLoSCapabilities.

### 4.4.6 Use cases

- Create class of service
- Create line of service
4.5 File system storage management domain

FileSystems provide access to byte-accessible storage through file-based protocols. This domain includes the management of resources that provide file-based access to storage.

File-based storage is represented by a FileSystem resources. Remote access to portions of a FileSystem is provided by FileShare resources.

4.5.1 Use cases

- Create file share
- Create file system
5 User Guidance

Swordfish supports a range of possible features. Clients use the Features registry to determine what SupportedFeatures a specific instance of an implementation is capable of. These range from discovery (read-only functionality), to event notification, to performance instrumentation, to multiple levels of block and file provisioning capabilities.

Supporting the granular feature definition is a detailed profile description that includes precise definitions of what functionality must be implemented in order to advertise support for each feature.

For use cases specified in this document, there is an expectation that specific features have been implemented to support the corresponding use case. For example, the “AssignReplicaTarget” use case highlights functionality for either local or remote replication, depending on the selected target volume’s system (either the same or different than the source volume). In either case, the client is working with the presumption that the feature has been advertised.
6 Features

6.1 Overview

This guide covers use cases for both installations that have opted to deploy a basic level of the API, and only extends the Redfish standard slightly, and those that implement a number of advanced features, providing a broader range of functionality. Use cases that assume functionality or features beyond a basic Swordfish implementation are clearly identified below, and are group with one another.

This version of the Users' Guide incorporates the functionality defined in v1.1.0 of the Swordfish specification, which defines a number of features. In addition to basic use cases, this document includes cases that rely on the implementation of each of those features:

- Block capacity management
- Block I/O performance
- Block provisioning
- Discovery
- EnergyStar
- Event notification
- File provisioning
- File capacity management
- Mapping and masking
- NVMe
- Replication (both local and remote)
6.2 Block provisioning feature

6.2.1 Overview

This feature provides basic, block-based storage provisioning.

6.2.2 Feature-specific Guidance

None defined.

6.2.3 Block provisioning use cases

The use cases defined for block provisioning are summarized in Table 8.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Drives to a StoragePool</td>
<td>Add multiple drives to an existing StoragePool</td>
</tr>
<tr>
<td>Create Volume from a StoragePool</td>
<td>Create Volume from an Existing Storage Pool</td>
</tr>
<tr>
<td>Create Volume using the Volume Collection</td>
<td>Create Volume using the Volume Collection</td>
</tr>
<tr>
<td>Delete Drives from a StoragePool</td>
<td>Add multiple drives from an existing StoragePool</td>
</tr>
</tbody>
</table>

6.3 Capacity management feature

6.3.1 Overview

This feature provides the management and alteration of storage once it has been allocated.

6.3.2 Feature-specific Guidance

None defined.

6.3.3 Capacity management use cases

The use cases defined for capacity management are summarized in Table 9.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expand Volume</td>
<td>Expand capacity of a Volume</td>
</tr>
</tbody>
</table>
6.4.1 Overview

This feature provides support for automated storage management based on ClassOfService and LineOfService modeling.

6.4.2 Feature-specific Guidance

6.4.2.1 Default Class of Service

If a pool, storage volume or other construct is created with no specified class of service when a class of service exists, the implementation will attempt to apply the DefaultClassOfService.

6.4.3 Class of service use cases

The use cases defined for class of service are summarized in Table 10.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create class of service</td>
<td>Create a class of service</td>
</tr>
<tr>
<td>Create file system</td>
<td>Create a file system</td>
</tr>
<tr>
<td>Create line of service</td>
<td>Create a line of service</td>
</tr>
<tr>
<td>Create Volume specifying Class of Service</td>
<td>Create Volume specifying Class of Service</td>
</tr>
<tr>
<td>Create Volume using Default Class of Service</td>
<td>Create Volume using Default Class of Service</td>
</tr>
<tr>
<td>Create an on-demand snapshot of a Volume</td>
<td>Create an on-demand snapshot of a Volume</td>
</tr>
</tbody>
</table>

6.5 Event notification feature

6.5.1 Overview

This feature provides basic eventing.

6.5.2 Feature-specific Guidance

None defined.

6.5.3 Event notification use cases

The use cases defined for event notification are summarized in Table 11.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscribe to thresholds events</td>
<td>Subscribe to Threshold Events</td>
</tr>
</tbody>
</table>
6.6 File provisioning feature

6.6.1 Overview

This feature provides file-based storage provisioning.

6.6.2 Feature-specific Guidance

None defined.

6.6.3 File provisioning use cases

The use cases defined for file provisioning are summarized in Table 12.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create file share</td>
<td>Create a file share</td>
</tr>
<tr>
<td>Create file system</td>
<td>Create a file system</td>
</tr>
</tbody>
</table>

6.7 Block IO performance feature

6.7.1 Overview

This feature provides basic performance metrics.

6.7.2 Feature-specific Guidance

None defined.

6.7.3 IO performance use cases

The use cases defined for IO performance are summarized in Table 13.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscribe to thresholds events</td>
<td>Subscribe to Threshold Events</td>
</tr>
</tbody>
</table>
6.8.3 Mapping and masking use cases

The use cases defined for mapping and masking are summarized in Table 14.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create StorageGroup</td>
<td>Create a StorageGroup</td>
</tr>
</tbody>
</table>

6.9 NVMe Support feature

6.9.1 Overview

This feature provides NVMe-specific use cases.

6.9.2 Feature-specific Guidance

The detailed mapping between Swordfish objects and the NVMe object model can be found in Swordfish NVMe Model Overview and Mapping Guide (https://www.snia.org/forums/smi/swordfish).

6.9.3 NVMe use cases

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach a Namespace</td>
<td>Attach a Namespace</td>
</tr>
<tr>
<td>Deprovision a Namespace</td>
<td>Deprovision a Namespace</td>
</tr>
<tr>
<td>Detach a Namespace</td>
<td>Detach a Namespace</td>
</tr>
<tr>
<td>Provision a Namespace</td>
<td>Provision a Namespace</td>
</tr>
<tr>
<td>Provision a Namespace with NVMSets</td>
<td>Provision a Namespace associated with an NVMSets</td>
</tr>
<tr>
<td>Report capacity for a Namespace</td>
<td>Report capacity for a Namespace</td>
</tr>
<tr>
<td>Report remaining life for a Namespace</td>
<td>Report remaining life for a Namespace</td>
</tr>
</tbody>
</table>
6.10 Replication Feature

6.10.1 Overview

This set of features (local replication and remote replication) provides support for a broad range of storage redundancy protections.

6.10.2 Feature-specific Guidance

Replication can be implemented many ways. The use cases defined for this feature illustrate two possible approaches, one using volumes and the other using consistency groups. Those use cases employing consistency groups will include "CG" in their titles, to avoid confusion.

6.10.3 Replication use cases

The use cases defined for replication are summarized in Table 15.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign Replica Target</td>
<td>Create a New Replication Relationship by Assigning a Target Volume</td>
</tr>
<tr>
<td>Assign Replica Target (CG)</td>
<td>Create a New Replication Relationship by Assigning an existing</td>
</tr>
<tr>
<td></td>
<td>Target Consistency Group</td>
</tr>
<tr>
<td>Create Replica Target</td>
<td>Make a New Replication Relationship by Creating a Target Volume</td>
</tr>
<tr>
<td>Create Replica Target (CG)</td>
<td>Make a New Replication Relationship by Creating a Target Consistency Group</td>
</tr>
<tr>
<td>Remove Replication Relationship</td>
<td>Remove Replication Relationship</td>
</tr>
<tr>
<td>Remove Replication Relationship (CG)</td>
<td>Remove Replication Relationship for a Consistency Group</td>
</tr>
<tr>
<td>Resume the Replication Synchronization</td>
<td>Resume the Replication Synchronization Activity</td>
</tr>
<tr>
<td>Activity</td>
<td></td>
</tr>
<tr>
<td>Resume the Replication Synchronization</td>
<td>Resume the Replication Synchronization Activity for a Consistency</td>
</tr>
<tr>
<td>Activity (CG)</td>
<td>Group</td>
</tr>
<tr>
<td>Reverse a Replication Relationship</td>
<td>Reverse a Replication Relationship</td>
</tr>
<tr>
<td>Reverse a Replication Relationship (CG)</td>
<td>Reverse a Replication Relationship for Consistency Groups</td>
</tr>
<tr>
<td>Split a Replica</td>
<td>Split a Replica</td>
</tr>
<tr>
<td>Split a Replica (CG)</td>
<td>Split a set of Replicas in Consistency Groups</td>
</tr>
<tr>
<td>Suspend Replication Synchronization Activity</td>
<td>Suspend Replication Synchronization Activity</td>
</tr>
<tr>
<td>Suspend Replication Synchronization Activity</td>
<td>Suspend Replication Synchronization Activity between Consistency</td>
</tr>
<tr>
<td></td>
<td>Groups</td>
</tr>
</tbody>
</table>
7 Use cases

7.1 Use cases by feature

7.1.1 Uncategorised Use Cases

The use cases that don't align with a particular feature are summarized in Table 16.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Storage Pool</td>
<td>Create a StoragePool</td>
</tr>
<tr>
<td>Create Storage Pool by drives</td>
<td>Create a StoragePool using a specific set of drives</td>
</tr>
<tr>
<td>Create Storage Pool with drives and RAIDType</td>
<td>Create storage pool using Specified Set of Drives and RAIDTypes</td>
</tr>
<tr>
<td>Subscribe to thresholds events</td>
<td>Subscribe to Threshold Events</td>
</tr>
</tbody>
</table>

7.1.2 Block provisioning use cases

The use cases defined for block provisioning are summarized in Table 17.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Drives to a StoragePool</td>
<td>Add multiple drives to an existing StoragePool</td>
</tr>
<tr>
<td>Create Volume from a StoragePool</td>
<td>Create Volume from an Existing Storage Pool</td>
</tr>
<tr>
<td>Create Volume using the Volume Collection</td>
<td>Create Volume using the Volume Collection</td>
</tr>
<tr>
<td>Delete Drives from a StoragePool</td>
<td>Add multiple drives from an existing StoragePool</td>
</tr>
</tbody>
</table>

7.1.3 Capacity management use cases

The use cases defined for capacity management are summarized in Table 18.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expand Volume</td>
<td>Expand capacity of a Volume</td>
</tr>
</tbody>
</table>

7.1.4 Class of service use cases

The use cases defined for class of service are summarized in Table 19.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
</table>
Create class of service
Create file system
Create line of service
Create Volume specifying Class of Service
Create Volume using Default Class of Service
Create an on-demand snapshot of a Volume

7.1.5 Event notification use cases
The use cases defined for event notification are summarized in Table 20.

Table 20: Event notification use cases

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscribe to thresholds events</td>
<td>Subscribe to Threshold Events</td>
</tr>
</tbody>
</table>

7.1.6 File provisioning use cases
The use cases defined for file provisioning are summarized in Table 21.

Table 21: File provisioning use cases

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create file share</td>
<td>Create a file share</td>
</tr>
<tr>
<td>Create file system</td>
<td>Create a file system</td>
</tr>
</tbody>
</table>

7.1.7 IO performance use cases
The use cases defined for IO performance are summarized in Table 22.

Table 22: IO performance use cases

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscribe to thresholds events</td>
<td>Subscribe to Threshold Events</td>
</tr>
</tbody>
</table>

7.1.8 Mapping and masking use cases
The use cases defined for mapping and masking are summarized in Table 23.

Table 23: Mapping and masking use cases

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create StorageGroup</td>
<td>Create a StorageGroup</td>
</tr>
</tbody>
</table>

7.1.9 NVMe use cases

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach a Namespace</td>
<td>Attach a Namespace</td>
</tr>
<tr>
<td>Deprovision a Namespace</td>
<td>Deprovision a Namespace</td>
</tr>
<tr>
<td>Detach a Namespace</td>
<td>Detach a Namespace</td>
</tr>
<tr>
<td>Provision a Namespace</td>
<td>Provision a Namespace</td>
</tr>
<tr>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Provision a Namespace with NVMSet</td>
<td>Provision a Namespace associated with an NVMSet</td>
</tr>
<tr>
<td>Report capacity for a Namespace</td>
<td>Report capacity for a Namespace</td>
</tr>
<tr>
<td>Report remaining life for a Namespace</td>
<td>Report remaining life for a Namespace</td>
</tr>
</tbody>
</table>
7.1.10 Replication use cases

The use cases defined for replication are summarized in Table 24.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign Replica Target</td>
<td>Create a New Replication Relationship by Assigning a Target Volume</td>
</tr>
<tr>
<td>Assign Replica Target (CG)</td>
<td>Create a New Replication Relationship by Assigning an existing Target Consistency Group</td>
</tr>
<tr>
<td>Create Replica Target</td>
<td>Make a New Replication Relationship by Creating a Target Volume</td>
</tr>
<tr>
<td>Create Replica Target (CG)</td>
<td>Make a New Replication Relationship by Creating a Target Consistency Group</td>
</tr>
<tr>
<td>Remove Replication Relationship</td>
<td>Remove Replication Relationship</td>
</tr>
<tr>
<td>Remove Replication Relationship (CG)</td>
<td>Remove Replication Relationship for a Consistency Group</td>
</tr>
<tr>
<td>Resume the Replication Synchronization Activity</td>
<td>Resume the Replication Synchronization Activity</td>
</tr>
<tr>
<td>Resume the Replication Synchronization Activity (CG)</td>
<td>Resume the Replication Synchronization Activity for a Consistency Group</td>
</tr>
<tr>
<td>Reverse a Replication Relationship</td>
<td>Reverse a Replication Relationship</td>
</tr>
<tr>
<td>Reverse a Replication Relationship (CG)</td>
<td>Reverse a Replication Relationship for Consistency Groups</td>
</tr>
<tr>
<td>Split a Replica</td>
<td>Split a Replica</td>
</tr>
<tr>
<td>Split a Replica (CG)</td>
<td>Split a set of Replicas in Consistency Groups</td>
</tr>
<tr>
<td>Suspend Replication Synchronization Activity</td>
<td>Suspend Replication Synchronization Activity</td>
</tr>
<tr>
<td>Suspend Replication Synchronization Activity (CG)</td>
<td>Suspend Replication Synchronization Activity between Consistency Groups</td>
</tr>
</tbody>
</table>
7.2 Alphabetic list of use cases

7.2.1 Create a New Replication Relationship by Assigning an existing Target Consistency Group

Summary: Establish a replication relationship by assigning an existing consistency group to serve as the target in the replication relationship for an existing consistency group.

Purpose: Leverage an existing consistency group resource to provide expanded data protection by creating a replication relationship with the specified source consistency group.

Who: StorageAdmin, DevOps

Management Domain: Block storage management

Triggers: Need to provide expanded data protection through a replica relationship with the specified source consistency group.

Detailed Context: The admin needs to provide expanded data protection for the specified source consistency group, and the configuration includes a pre-existing consistency group that can be repurposed as the replication target.

Preconditions: User has already identified an existing consistency group to use for the target replicas (including the target system and storage pool properties), the type of replica(s), and the replica update mode (sync vs async).

Feature(s): Replication (both local and remote)

Inputs:
- URL for target volume: `/redfish/v1/Storage/1/ConsistencyGroup/CG_DB2`
- Requested replica type: Mirror
- ReplicaUpdateMode: Synchronous

Basic Course of Events:

1. Post (as an Action) the request on the source ConsistencyGroup.

This instructs the service to use the identified ConsistencyGroup as the source ConsistencyGroup for the specified replication relationship. For any additional details required, the service will rely on default values.

Request: POST `/redfish/v1/Storage/1/ConsistencyGroup/CG_DB1/ConsistencyGroup.AssignReplicaTarget`

Headers: No additional headers required.

Body:

```
{
    "ReplicaUpdateMode": "Synchronous",
    "TargetConsistencyGroup": "/redfish/v1/Storage/1/ConsistencyGroup/CG_DB2",
    "ReplicaType": "Mirror"
}
```

- Response contains the details of the created volume.

Response:

Headers: Location = /redfish/v1/Storage/1/ConsistencyGroup/CG_DB1

Body:

```
{
    "@SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
    "@odata.context": "/redfish/v1/$metadata#ConsistencyGroup.ConsistencyGroup",
    "@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB1",
    "@odata.type": ">#ConsistencyGroup_1_0_0.ConsistencyGroup",
}
```
**Postconditions:** The selected ConsistencyGroup has an updated ReplicaTargets entry for the new relationship. Elsewhere, there is a ConsistencyGroup "CG_DB2" in the system that has a ReplicaInfo which points back to this ConsistencyGroup and which contains all of the Replica configuration information.

**Failure Scenario:** None defined

See also: Assign Replica Target.
7.2.2 Attach a Namespace

Summary: Attach a Namespace

Purpose: Provide visibility to a namespace by attaching it to an IO Controller.

Who: StorageAdmin, CloudAdmin

Management Domain: Block storage management

Triggers: None defined.

Detailed Context: Attach a Namespace to an IO Controller to make it visible to the hosts that connect to that IO Controller, and accessible for block storage operations.

Preconditions: The IO Controller and Namespaces need to exist, and be fully defined.

Feature(s): NMVe

Inputs:

- URL for Namespace.

Basic Course of Events:

1. PATCH the AttachedVolumes array in the IO Controller with the Namespace.

   - **Request**: PATCH /redfish/v1/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Controllers/NVMeIOController/

   - **Headers**: No additional headers required.

   - **Body**:

     ```json
     {
       "Links": {
         "AttachedVolumes": [
           {
             "@odata.id": "/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Volumes/SimpleNamespace"
           }
         ]
       }
     }
     ```

   - **Response**: None defined.

Postconditions: The Namespace has been configured as an AttachedVolume to the IO Controller. This means that the Namespace is visible to hosts connected to the IO Controller.

Failure Scenario: None defined.

See also: None defined.
### 7.2.3 Create an on-demand snapshot of a Volume

**Summary:** Create an on-demand snapshot of a volume

**Purpose:** Create a snapshot of an existing volume. The resulting snapshot can then be used either as a backup or as a separate copy to test against.

**Who:** CloudAdmin, StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need an on-demand snapshot of a Volume

**Detailed Context:** To take a snapshot of a volume, we need to find a DataProtectionLineOfService that can describe the new Snapshot. The DataProtectionLineOfService must have no schedule set to be able to provide on-demand behavior.

**Preconditions:** None.

**Feature(s):** Class of Service

**Inputs:**
- URL for storage service: /redfish/v1/StorageServices/1
- URL for the volume: /redfish/v1/StorageServices/1/StoragePools/BasePool/AllocatedVolumes/1

**Basic Course of Events:**

1. The first step is to get the current supported DataProtectionLineOfService entries, and find one that provides appropriate snapshot support (i.e., has no schedule set, and meets any other criteria the client may have).

   **Request:**
   ```
   GET /redfish/v1/StorageServices/1/Links/DataProtectionLoSCapabilities/SupportedLinesOfService
   ```

   **Headers:**
   No additional headers required.

   **Body:**
   ```
   {   "SupportedLinesOfService": [   {   "@odata.id": "/redfish/v1/StorageServices/1/Links/DataProtectionLoSCapabilities/SupportedLinesOfService/OnDemandSnapshot",   "Name": "OnDemandSnapshot",   "RecoveryGeographicObjective": "Server",   "ReplicaType": "Snapshot",   }   ] }
   ```

   2. This looks like the right line of service for an on-demand snap. Note that since this is an on-demand snapshot, it has no schedule specified. So, let's use this to create the snapshot. We'll copy by value.

   **Request:**
   ```
   POST /redfish/v1/StorageServices/1/Links/DataProtectionLoSCapabilities/SupportedLinesOfService/OnDemandSnapshot.CreateReplicas
   ```

   **Headers:**
   No additional headers required.

   **Body:**
   ```
   {   "ReplicaRequests": [   {   "ReplicaSource": {"@odata.id": "/redfish/v1/StorageServices/1/StoragePools/BasePool/AllocatedVolumes/1"},   "ReplicaName": "MySnapshot"   }   ] }
   ```

   **Response:**
200 Success

• **Headers:**
  
  No additional headers required.

• **Body:**

```json
{
   "ReplicaPriority": "Same",
   "ReplicaReadOnlyAccess": "ReplicaElement",
   "UndiscoveredElement": null,
   "WhenSynced": "20171024T142000-00500",
   "SyncMaintained": null,
   "ReplicaRecoveryMode": null,
   "ReplicaUpdateMode": "Synchronous",
   "PercentSynced": 100,
   "FailedCopyStopsHostIO": null,
   "WhenActivated": "20171024T142000-00500",
   "WhenDeactivated": null,
   "WhenEstablished": "20171024T142000-00500",
   "WhenSuspended": null,
   "WhenSynchronized": null,
   "ReplicaSkewBytes": null,
   "ReplicaType": "Snapshot",
   "ReplicaProgressStatus": "Completed",
   "ReplicaState": "Synchronized",
   "ConsistencyEnabled": false,
   "ConsistencyType": null,
   "ConsistencyState": null,
   "ConsistencyStatus": null,
   "ReplicaRole": "Target",
   "Replica": {"odata.id": "/redfish/v1/StorageServices/1/StoragePools/BasePool/AllocatedVolumes/MySnapshot"},
   "DataProtectionLineOfService": "/redfish/v1/StorageServices/1/Links/DataProtectionLoSCapabilities/SupportedLinesOfService/OnDemandSnapshot"
}
```

**Postconditions:**

The original Volume (/redfish/v1/StorageServices/1/StoragePools/BasePool/AllocatedVolumes/1) now has a snapshot volume (/redfish/v1/StorageServices/1/StoragePools/BasePool/AllocatedVolumes/MySnapshot) that is available through its ReplicaTargets collection. All replication information (ReplicaInfo) is available on the snapshot (/redfish/v1/StorageServices/1/Volumes/MySnapshot) volume to describe the relationship.

**Failure Scenario:**

None defined.

**See also:**

None defined.
7.2.4 Create class of service

**Summary:** Create a class of service

**Purpose:** Create a new class of service in the service catalog to match a newly available type of storage

**Who:** StorageAdmin

**Management Domain:** Block storage management, Service catalog management

**Triggers:** The administrator has determined that a new class of service needs to be created to reflect a new class of SSD storage in the infrastructure.

**Detailed Context:** This is a simple scenario where the primary characteristic is the enhanced performance available of SSD drives.

** Preconditions:** None defined.

**Feature(s):** Class of Service

**Inputs:**

- URL for Storage Service: `/redfish/v1/StorageServices(1)`
- New class of service characteristics
  - Name: "SSD"
  - Description: "Minimal SSD class of service."
  - IOPerformanceLineOfService
    - "Name": "SSDLoS"
    - "IOOperationsPerSecondIsLimited": false
    - "MaxIOOperationsPerSecondPerTerabyte": 100000
    - "AverageIOPerOperationLatencyMicroseconds": 10

**Basic Course of Events:**

1. Create ClassOfService

   - **Request:** POST `/redfish/v1/StorageServices(1)/ClassesOfService/Members`

     - **Headers:** No additional headers required.

     - **Body:**

       ```
       {
         "Name": "SSD",
         "Description": "Minimal SSD class of service.",
         "LinesOfService": {
           "IOPerformanceLinesOfService": [
             {
               "Name": "SSDLoS",
               "IOOperationsPerSecondIsLimited": false,
               "MaxIOOperationsPerSecondPerTerabyte": 100000,
               "AverageIOPerOperationLatencyMicroseconds": 10
             }
           ]
         }
       }
       ```

   2. Response:

      - **Headers:**

        - Location = `/redfish/v1/StorageServices(1)/ClassesOfService(SSD)`

   **Postconditions:** The requested class of service is added to the ClassesOfService collection.
**Failure Scenario:** None defined.

**See also:** None defined.
7.2.5 Create file share

Summary: Create a file share

Purpose: Share an existing file system as /Shares/MyShare

Who: CloudAdmin, StorageAdmin, DevOps

Management Domain: File system storage management

Triggers: None defined.

Detailed Context: Create a share starting at /Shares/MyShare.

Preconditions: None defined.

Feature(s): File provisioning

Inputs:
- URL for the filesystem: /redfish/v1/StorageServices/1/FileSystems/QuickFiles
- The path to the shared file: "/Shares/MyShare"
- Description: "Share of files under MyShare."

Basic Course of Events:

1. Create a file share

   - Request: POST /redfish/v1/StorageServices/1/FileSystems/QuickFiles/ExportedShares

   - Headers: No additional headers required.

   - Body:

   ```json
   {
     "Name": "MyShare",
     "Description": "Share of files under MyShare."
   }
   ```

   - Response:

   - Headers:

   Location= /redfish/v1/StorageServices/1/FileSystems/QuickFiles/ExportedShares/MyShare

Postconditions: The requested file share is added to the ExportedShares collection for the file system.

Failure Scenario: None defined.

See also: None defined.
7.2.6 Create file system

**Summary:** Create a file system

**Purpose:** Create a file system with a given capacity and performance level.

**Who:** StorageAdmin

**Management Domain:** File system storage management

**Triggers:** None defined.

**Detailed Context:** Create a 100 TB file system based on SSD class storage.

**Preconditions:** None defined.

**Feature(s):** Class of Service

**Inputs:**
- URL for storage service: `/redfish/v1/StorageServices/1`
- Name for the new file system: “QuickFiles”
- Description: "100 TB FileSystem having SSD class storage."
- URL for class of service: `/redfish/v1/StorageServices/1/Links/ClassesOfService/SSD`
- File system capacity:

```
{
    "Data": {
        "ProvisionedBytes": 100000000000000,
        "IsThin Provisioned": true
    }
}
```

**Basic Course of Events:**

1. Create FileSystem

   **Request:** POST /redfish/v1/StorageServices/1/FileSystems

   **Headers:** No additional headers required.

   **Body:**

   ```json
   {
       "Name": "QuickFiles",
       "Description": "100 TB FileSystem having SSD class storage.",
       "Capacity": {
           "Data": {
               "ProvisionedBytes": 100000000000000,
               "IsThin Provisioned": true
           }
        },
       "Links": {
           "ClassOfService": {
               "odata.id": "/redfish/v1/StorageServices/1/Links/ClassesOfService/SSD"
           }
        }
   }
   ```

   **Response:**
* **Headers:**

  Location= /redfish/v1/StorageServices/1/FileSystems/QuickFiles

**Postconditions:** The requested file system is added to the FileSystems collection for the Storage Service.

**Failure Scenario:** None defined.

**See also:** None defined.
7.2.7 Create file system

Summary: Create a file system

Purpose: Create a file system with a given capacity and performance level.

Who: StorageAdmin

Management Domain: File system storage management

Triggers: None defined.

Detailed Context: Create a 100 TB file system based on SSD class storage.

Preconditions: None defined.

Feature(s): File provisioning

Inputs:

- URL for storage service: /redfish/v1/StorageServices/1
- Name for the new file system: “QuickFiles”
- Description: "100 TB FileSystem having SSD class storage."
- URL for class of service: /redfish/v1/StorageServices/1/Links/ClassesOfService/SSD
- File system capacity:

```
{
    "Data": {
        "ProvisionedBytes": 100000000000000,
        "IsThinProvisioned": true
    }
}
```

Basic Course of Events:

1. Create FileSystem

   • **Request**: POST /redfish/v1/StorageServices/1/Filesystems

     • **Headers**: No additional headers required.

     • **Body**:

```
{
    "Name": "QuickFiles",
    "Description": "100 TB FileSystem having SSD class storage.",
    "Capacity": {
        "Data": {
            "ProvisionedBytes": 100000000000000000
        },
        "IsThinProvisioned": true
    },
    "Links": {
        "ClassOfService": {"odata.id": "/redfish/v1/StorageServices/1/Links/ClassesOfService/SSD"
    }
}
```

• **Response**:
* **Headers:**
  Location= /redfish/v1/StorageServices/1/FileSystems/QuickFiles

**Postconditions:** The requested file system is added to the FileSystems collection for the Storage Service.

**Failure Scenario:** None defined.

**See also:** None defined.
7.2.8 Create line of service

Summary: Create a line of service to reflect the performance characteristics of SSD storage

Purpose: The definition is created here in preparation of creating ClassOfService instances that include a requirement for SSD storage performance.

Who: StorageAdmin

Management Domain: Block storage management, Service catalog management

Triggers: None defined.

Detailed Context: SSD storage is introduced and need a new performance line of service to reflect their capability.

Preconditions: None defined.

Feature(s): Class of Service

Inputs:
- URL for Storage Service: /redfish/v1/StorageServices/1
- New IO performance line of service

```
{
  "Name": "NewSSDLoS",
  "IoOperationsPerSecondIsLimitedBoolean": false,
  "MaxIoOperationsPerSecondPerTerabyte": 100000,
  "AverageIoOperationLatencyMicroseconds": 10
}
```

Basic Course of Events:

1. Get existing supported lines of service

   - Request:
     GET /redfish/v1/StorageServices/1/Links/IOPerformanceLoSCapabilities/SupportedIOPerformanceLinesOfService

   - Headers: No additional headers required.

   - Reply:
     - Headers: `ETag: 123-a`

     - Body:

```
[
  {
    "Name": "LoS1",
    "IoOperationsPerSecondIsLimitedBoolean": false,
    "SamplePeriodSeconds": 60,
    "MaxIoOperationsPerSecondPerTerabyte": 83,
    "AverageIoOperationLatencyMicroseconds": 8000,
  },
  {
    "Name": "LoS2",
    "IoOperationsPerSecondIsLimitedBoolean": false,
    "SamplePeriodSeconds": 60,
    "MaxIoOperationsPerSecondPerTerabyte": 133,
    "AverageIoOperationLatencyMicroseconds": 5000,
    "IOWorkload": {
      "Name": "Duplicon: OLTP"
    }
  }
]`
2. Create new line of service

- **Request**:

  PATCH /redfish/v1/StorageServices/1/Links/IOPerformanceLoSCapabilities

- **Headers**:

  - If-Match: "123-a"

- **Body**:

  ```json
  {
    "SupportedIOPerformanceLinesOfService": [
      {
        "Name": "LoS1",
        "IoOperationsPerSecondIsLimitedBoolean": false,
        "SamplePeriodSeconds": 60,
        "MaxIoOperationsPerSecondPerTerabyte": 83,
        "AverageIoOperationLatencyMicroseconds": 8000
      },
      {
        "Name": "LoS2",
        "IoOperationsPerSecondIsLimitedBoolean": false,
        "SamplePeriodSeconds": 60,
        "MaxIoOperationsPerSecondPerTerabyte": 133,
        "AverageIoOperationLatencyMicroseconds": 5000,
        "IOWorkload": {
          "Name": "Duplicon: OLTP"
        }
      },
      {
        "Name": "NewSSDLoS",
        "IoOperationsPerSecondIsLimitedBoolean": false,
        "MaxIoOperationsPerSecondPerTerabyte": 100000,
        "AverageIoOperationLatencyMicroseconds": 10
      }
    ]
  }
  ```

- **Response**:

  - **Headers**: None defined.
  - **Body**: None defined.

**Postconditions**: The requested line of service is added to the SupportedIOPerformanceLinesOfService of the Storage Service.

**Failure Scenario**: None defined.

**See also**: None defined.
7.2.9 Create storage pool using Specified Set of Drives and RAIDTypes

Summary: Create a StoragePool

Purpose: Create a StoragePool using a Specified set of drives and a specified set of supported RAIDTypes

Who: StorageAdmin

Management Domain: Block storage management

Triggers: Users need to allocate storage.

Detailed Context: Create a storage pool containing an amount of storage that can be used to create volumes with the specified set of RAIDTypes.

Preconditions: The user has the information about the set of drives.

Feature(s): None.

Inputs:

- URL for storage pool collection:
  - NSB: /redfish/v1/Storage/1/StoragePool
  - (Also applicable to SB: /redfish/v1/StorageServices/1/StoragePool)
- Name for the new storage pool: "SSD"
- Description: "SSD Storage Pool"
- Set of Drives:

```
{
   "Drives": [
   {
   "odata.id": "redfish/v1/Storage/1/Drive/1"},
   {
   "odata.id": "redfish/v1/Storage/1/Drive/2"},
   {
   "odata.id": "redfish/v1/Storage/1/Drive/3"},
   {
   "odata.id": "redfish/v1/Storage/1/Drive/4"},
   {
   "odata.id": "redfish/v1/Storage/1/Drive/5"},
   {
   "odata.id": "redfish/v1/Storage/1/Drive/6"},
   {
   "odata.id": "redfish/v1/Storage/1/Drive/7"},
   {
   "odata.id": "redfish/v1/Storage/1/Drive/8"}
   ]
}
```

- Set of Supported RAIDTypes:

```
{
   "SupportedRAIDTypes": ["RAID1","RAID5","RAID50"]
}
```

Basic Course of Events:

1. Create StoragePool

   - Request: POST /redfish/v1/Storage/1/StoragePools/

     - Headers: No additional headers required.

     - Body:

```
{
   "Name": "SSD",
   "Description": "Storage Pool for RAID1, 5 or 50.",
   "RAIDTypes": ["RAID1","RAID5","RAID50"]
}
```
"Drives": [
  {"odata.id": "/redfish/v1/Storage/1/Drive/1"},
  {"odata.id": "/redfish/v1/Storage/1/Drive/2"},
  {"odata.id": "/redfish/v1/Storage/1/Drive/3"},
  {"odata.id": "/redfish/v1/Storage/1/Drive/4"},
  {"odata.id": "/redfish/v1/Storage/1/Drive/5"},
  {"odata.id": "/redfish/v1/Storage/1/Drive/6"},
  {"odata.id": "/redfish/v1/Storage/1/Drive/7"},
  {"odata.id": "/redfish/v1/Storage/1/Drive/8"}
]

* Response: None.

* Headers:

Location=/redfish/v1/Storage/1/StoragePools/SSD

**Post-Conditions:** The requested StoragePool is added to the collection for Storage.

**Failure Scenario:** None defined.

**See also:** None defined.
7.2.10 Create storage pool using Specified Set of Drives

Summary: Create a StoragePool

Purpose: Create a StoragePool using a specified set of drives

Who: StorageAdmin

Management Domain: Block storage management

Triggers: Users need to allocate storage.

Detailed Context: Create a storage pool containing an amount of storage that can be used to create volumes with the specified set of RAIDTypes.

Preconditions: The user has the information about the set of drives.

Feature(s): None.

Inputs:

- URL for storage pool collection:
  - NSB: /redfish/v1/Storage/1/StoragePool
  - (Also applicable to SB: /redfish/v1/StorageServices/1/StoragePool)

- Name for the new storage pool: "SP1"

- Description: "Storage Pool 1"

- Set of Drives:

  ```json
  { 
    "Drives": [ 
      {"odata.id": "/redfish/v1/Storage/1/Drive/1"} 
    ]
  }
  ```

Basic Course of Events:

1. Create StoragePool

- Request: POST /redfish/v1/Storage/1/StoragePools/

- Headers: No additional headers required.

- Body:

  ```json
  { 
    "Name": "SSD",
    "Description": "Storage Pool 1",
    "Drives": [ 
      {"odata.id": "/redfish/v1/Storage/1/Drive/1"}
    ]
  }
  ```

- Response:

- Headers:

  ```text
  Location=/redfish/v1/Storage/1/StoragePools/SP1`
  ```

Post-Conditions: The requested StoragePool is added to the collection for Storage.

Failure Scenario: None defined.

See also: None defined.
7.2.11 Create storage pool

Summary: Create a StoragePool

Purpose: Create a StoragePool

Who: StorageAdmin

Management Domain: Block storage management

Triggers: Users need to allocate storage with characteristics satisfied by a class of service.

Detailed Context: Create a storage pool containing an amount of storage that can be used to create a requested class of service. The storage pool implementation will attempt to find and allocate enough storage that will satisfy the request. No metadata or snapshot storage is reserved.

Preconditions: None defined.

Feature(s): Class of Service

Inputs:

- URL for storage service: /redfish/v1/StorageServices/1
- Name for the new storage pool: "SSD"
- Description: "100 TB pool of SSD class storage."
- URL for class of service: /redfish/v1/StorageServices/1/ClassesOfService/SSD
- Storage pool capacity

```json
{
   "Data": {
      "ProvisionedBytes": 100000000000000,
      "IsThinProvisioned": false
   }
}
```

Basic Course of Events:

1. Create StoragePool

   - Request: POST /redfish/v1/StorageServices/1/StoragePools/Members

     - Headers: No additional headers required.

     - Body:

       ```json
       {
         "Name": "SSD",
         "Description": "100 TB pool of SSD class storage.",
         "Capacity": {
            "Data": {
               "ProvisionedBytes": 100000000000000
            },
            "IsThinProvisioned": false
         },
         "ClassesOfService": {
            "Members": [
               {
                  "@odata.id": "/redfish/v1/StorageServices/Members/1/ClassesOfService/SSD"
               }
            ]
         }
       }
       ```

   - Response:

     - Headers:
Location=/redfish/v1/StorageServices/1/StoragePools/SSD

**Post-Conditions:** The requested StoragePool is added to the collection for the Storage Service.

**Failure Scenario:** None defined.

**See also:** None defined.
7.2.12 Create StorageGroup

Summary: Create a StorageGroup

Purpose: Create a StorageGroup

Who: CloudAdmin

Management Domain: Block storage management

Triggers: None defined.

Detailed Context: Create a collection of application storage that is exposed to an application and managed as a unit.

Preconditions: None defined.

Feature(s): Mapping and masking

Inputs:
- URL for storage service: /redfish/v1/StorageService
- The proposed name of the new StorageGroup (string): "HillOfBeans1"
- A description of the new StorageGroup (string): "New storage group to support accounting"
- A consistency requirement for the new StorageGroup (boolean). When TRUE, all copies of data within a StorageGroup must be kept in sync: true

Basic Course of Events:

1. Create StorageGroup

   * Request: POST /redfish/v1/StorageService/StorageGroups/Members

     * Headers: No additional headers required.

     * Body:

     ```json
     {}
     "Name" : "HillOfBeans1",
     "Description" : "New storage group to support accounting",
     "MembersAreConsistent" : true
     ```

   * Response: None defined.

Postconditions: The requested StorageGroup is added to the Members collection for the Storage Service.

Failure Scenario: None defined.

See also: None defined.
7.2.13 Create Volume from an Existing Storage Pool

**Summary:** Create a new Volume from a Storage Pool.

**Purpose:** Create a new volume, with a specified capacity, from a previously created StoragePool.

**Who:** CloudAdmin, StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to allocate storage for a new application.

**Detailed Context:** The admin needs to satisfy a user or application request to provide a given amount of capacity with a specified RAIDType, and wants to leverage a preconfigured Storage Pool.

**Preconditions:** User has already selected a Pool and checked the supported RAIDType properties.

**Feature(s):** Block provisioning

**Inputs:**
- URL for Storage Pool: /redfish/v1/Storage/1/StoragePools/PrimaryPool
- Requested volume size in bytes (1 TiB, here): 1099511627776
- Requested name of volume: "MyVolume"
- RAIDType: RAID1

**Basic Course of Events:**

1. Post the definition of the new volume to the Volumes resource collection.

This instructs the service to use the identified StoragePool to allocate a new volume of the requested size that meets the requirements of the specified protection level. Since additional details are not provided, the service will rely on default values as required.

**Request:** POST /redfish/v1/Storage/1/StoragePools/PrimaryPool

**Headers:** No additional headers required.

**Body:**

```
{
    "Name": "MyVolume",
    "CapacityBytes": 1099511627776,
    "RAIDType": "RAID1"
}
```

* Response contains the details of the created volume.

**Response:**

**Headers:** Location = /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1

**Body:**

```
{
  "@@SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
  "@odata.context": "/redfish/v1/$metadata#Volume.Volume",
  "@odata.id": "/redfish/v1/Storage/1/Volumes/1",
  "@odata.type": ">#Volume_1_6_0.Volume",
  "Name": "MyVolume",
  "Id": "1",
  "Description": "Default Volume Description",
  "RAIDType": "RAID1",
  "Identifiers": [}
```
Postconditions: The selected volume is added to the AllocatedVolumes collection within the selected storage pool.

Failure Scenario: None defined

See also: None defined.
7.2.14 Create Volume using the Volume Collection

**Summary:** Add a new Volume to the Volume Collection

**Purpose:** Create a new volume with a specified capacity, RAID Type, and set of drives.

**Who:** CloudAdmin, StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to allocate storage for a new application.

**Detailed Context:** The admin needs to satisfy a user or application request to provide a given amount of capacity, and to assure a given protection level.

**Preconditions:** None.

**Feature(s):** Block provisioning

**Inputs:**
- URL for Volume Collection: /redfish/v1/Storage/1/Volumes
- Requested volume size in bytes (1 TiB, here): 1099511627776
- Requested name of volume: "MyVolume"
- RAIDType: RAID1
- Drives: Two drives (to satisfy the requested RAID level) from /redfish/v1/Storage/1/Drives: - /redfish/v1/Storage/1/Drive/1 - /redfish/v1/Storage/1/Drive/2

**Basic Course of Events:**

1. Post the definition of the new volume to the Volumes resource collection.

This instructs the service to allocate a new volume of the requested size that meets the requirements of the specified protection level. Since additional details are not provided, the service will rely on default values as required.

**Request:** POST /redfish/v1/Storage/1/Volumes

**Headers:** No additional headers required.

**Body:**

```json
{
  "Name": "MyVolume",
  "CapacityBytes": 1099511627776,
  "RAIDType": "RAID1",
  "Drives": [
    {"odata.id": "/redfish/v1/Storage/1/Drive/1"},
    {"odata.id": "/redfish/v1/Storage/1/Drive/2"}
  ]
}
```

- Response contains the details of the created volume.

**Response:**

**Headers:** Location = /redfish/v1/Storage/1/Volumes/1

**Body:**

```json
{
  "@SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved."
}
```
Postconditions: The selected volume is added to the Volume collection for Storage.

Failure Scenario: None defined

See also: None defined.
7.2.15 Create Volume specifying Class of Service

**Summary:** Create a Volume

**Purpose:** Create a Volume with a known capacity and class of service.

**Who:** CloudAdmin, StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to allocate storage for a new application.

**Detailed Context:** The admin needs to satisfy a service request to provide a given amount of storage to an application, and to assure a given class of service.

**Preconditions:** None.

**Feature(s):** Class of Service

**Inputs:**
- URL for Storage Service: `/redfish/v1/StorageServices/1`
- Requested volume size in bytes (for 1TiB in this example): 1099511627776
- URL for requested class of service: `/redfish/v1/StorageServices/1/ClassesofService/BostonBunker`
- Requested name of volume (string): "Snapshot1"

**Basic Course of Events:**
1. Post the definition of the new volume to the Volumes resource collection.

   This instructs the service to allocate a new volume of the requested size that meets the requirements of the specified class of service. Since additional details are not provided, the service is free to allocate the storage from any of its storage pools that can satisfy the request.

   - **Request:** POST `/redfish/v1/StorageServices/1/Volumes/Members`
   - **Headers:** No additional headers required.
   - **Body:**

     ```json
     {
         "Name": "Snapshot1",
         "CapacityBytes": 1099511627776,
         "Links": {
             "ClassOfService": {
                 "odata.id": "/redfish/v1/StorageServices/1/ClassesofService/BostonBunker"
             }
         }
     }
     ```

   - **Response:**
   - **Headers:**

     Location = `/redfish/v1/StorageServices/1/Volumes/3`

     ```json
     {
         "@SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
         "@odata.context": "/redfish/v1/$metadata#Volume.Volume",
         "@odata.id": "/redfish/v1/StorageServices/1/Volumes/3",
         "@odata.type": ">
```
Postconditions: The selected volumes are added to the collection for the Storage Group.

Failure Scenario: None defined
See also: None defined.
7.2.16 Deprovision a Namespace

Summary: Deprovision a Namespace

Purpose: Deprovision (delete) a Namespace that is no longer needed.

Who: StorageAdmin CloudAdmin

Management Domain: Block storage management

Triggers: None defined.

Detailed Context: Delete the namespace and free resources reserved for it. This can happen regardless of the controller attachment state of the namespace.

Preconditions: None defined.

Feature(s): NVMe

Inputs:
- URL for the namespace, either within the containing NVM set (a storage pool) or directly from the containing subsystem.

Basic Course of Events:

1. Delete the Namespace.
   - Request: DELETE /redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Volumes/SimpleNamespace
   - Headers: No additional headers required.
   - Body: None defined.
   - Response: None defined.

Postconditions: The requested namespace has been deprovisioned.

Failure Scenario: None defined.

See also: None defined.
Expand capacity of a storage volume

**Summary:** Retrieve storage capacity information for a storage pool to check available capacity before expanding a storage volume.

**Purpose:**
- Application monitoring tool warns the admin that a storage volume used by a critical application is at 80% capacity
- Storage volumes can be expanded during the week without approval during non-business hours
- Administrator checks the available capacity on the storage pool based on the class of service required to ensure the storage volume expansion is possible
- During proper maintenance window, administrator expands storage volume based on predefined SLA established with application owner

**Who:** StorageAdmin at an enterprise

**Management domain:** Block storage management

**Trigger:** Low available capacity warning

**Detailed context:** The enterprise administrator offers managed storage services to business units in her organization. The administrator has established an SLA with application owners for storage volume expansion:
- **Low** storage capacity warning and critical alerts only apply to storage volumes with application data (the storage administrator is not responsible for the OS disks of the servers)
- Administrator can expand the storage volume by 25% up to 3 times without additional approvals for a maintenance window as long as the work is done during non-business hours (business hours are 7AM - 7PM local).
- Storage volumes for this application cannot exceed 2TB.
- Storage volume will be tagged with metadata to indicate the original size of the volume.
- Administrator is responsible for informing the application owner if the storage system is running low and overall capacity putting the SLA at risk. Administrator and application owner must plan for a migration of the application if waiting for additional storage disks/shelves to arrive is not feasible.

The application monitoring tool sends a warning alert to the administrator’s datacenter monitoring tool indicating that a storage volume’s capacity is at 80% and must be increased to ensure the application does not experience unplanned downtime. The administrator first checks the available capacity on the storage pool for a given storage class of service. Since the storage system is new, there is sufficient capacity available. Next the administrator figures out that the storage volume has never been expanded based on the original size tag. At the start of the maintenance window, non-business hours, the administrator initiates the expand action. For this particular storage system, the expand is a long running action so the administrator tracks the progress using the associated task.

**Preconditions:**
- Storage system has at least one storage pool with at least one storage volume
- Storage pool has enough available capacity to expand the storage volume by 25%

**Feature(s):** Block capacity management

**Inputs:**
- URL for the Swordfish service: `/redfish/v1/StorageService/1`
- Storage pool name: BasePool
- Storage volume name: 61001234876545676100123487654567

**Basic Course of Events:**

1. Uses GET operations to look at information about storage volume and confirm low capacity and the current size is less than 2TB

   **Request:**
   ```
   GET /redfish/v1/StorageServices/1/Volumes/61001234876545676100123487654567
   ```

   **Response:**
   ```
   "@Redfish.Copyright": "Copyright 2015-2021 SNIA. All rights reserved.",
   "@odata.context": "/redfish/v1/$metadata#Volume.Volume",
   "@odata.id": "/redfish/v1/StorageServices/1/Volumes/61001234876545676100123487654567",
   "@odata.type": ">#Volume_1_0_0.Volume",
   "Id": "61001234876545676100123487654567",
   "Name": "Volume 1",
   ```
2. Uses GET operations to look at information about storage pool to confirm it has more than 25% available capacity

- **Request:** GET /redfish/v1/StorageServices/1/StoragePools/BasePool

- **Response:**

```json

{  
  "Description": "application storage",  
  "Identifiers": [  
    {  
      "DurableNameFormat": "NAA6",  
      "DurableName": "610012347654567610012347654567"  
    }  
  ],  
  "Manufacturer": "SuperDuperStorageProvider",  
  "Status": {  
    "State": "Enabled",  
    "Health": "OK"  
  },  
  "BlockSizeBytes": 512,  
  "LowSpaceWarningThresholdPercent": [  
    80,  
    null,  
    null,  
    null,  
    null  
  ],  
  "Capacity": {  
    "Data": {  
      "ConsumedBytes": 879609302221,  
      "AllocatedBytes": 879609302221,  
      "GuaranteedBytes": 549755813888,  
      "ProvisionedBytes": 1099511627776  
    },  
    "Metadata": null,  
    "Snapshot": null  
  }  
}
```
3. Use expand action to increase size of storage volume by 25%.

```
"CapacitySources": [

"ProvidedCapacity": {
  "ConsumedBytes": 70368744177664,
  "AllocatedBytes": 140737488355328,
  "GuaranteedBytes": 17592186044416,
  "ProvisionedBytes": 56294953421312
},

"Links": {
  "ClassOfService": {
    "@odata.id": "/redfish/v1/StorageServices/1/ClassesOfService/GoldBoston"
  },
  "ProvidingPool": null,
  "ProvidingVolume": null
},

"LowSpaceWarningThresholdPercent": [70, 80, 90],

"Links": {
  "AllocatedPools": [],
  "AllocatedVolumes": [
    {
      "@odata.id": "/redfish/v1/StorageServices/1/Volumes/61001234876545676100123487654567"
    },
  ],
  "SupportedClassesOfService": [
    {
      "@odata.id": "/redfish/v1/StorageServices/1/ClassesOfService/GoldBoston"
    },
    {
      "@odata.id": "/redfish/v1/StorageServices/1/ClassesOfService/SilverBoston"
    }
  ]
]
```

Note that there are two different properties that can be used to report available capacity, depending on the implementation - either CapacityBytes or Capacity->Data->AllocatedBytes. This use case shows how to do this function using the latter property, but would work equally if the implementation supported this function using the CapacityBytes property instead.

**Request:** `PATCH /redfish/v1/StorageServices/1/Volumes/61001234876545676100123487654567`

**Headers:** No additional headers required

**Body:**

```
{
  "Capacity": {
    "Data": {
      "AllocatedBytes": 1374389534720
    }
  }
}
```
* Response:

```json
{
    "taskid": "ExpandTask123"
}
```

**Postconditions:** The volume's capacity expands by 25%. Administrator needs to track the associated task to know when the volume expansion completes.

**See also:** None defined.
## 7.2.18 Health updates

**Summary:** Retrieve storage metrics information for a storage volume.

**Purpose:**
- Application monitoring tool warns devops that the application is processing less requests based on a predefined threshold
- Devops has a simple script that queries Redfish capable infrastructure and pulls performance metrics
- Devops collect infrastructure performance metrics as part of an end-to-end diagnostics workflow to help identify potential bottlenecks

**Who:** DevOps at an enterprise

**Management domain:** Block storage management

**Trigger:** Lower than expected application requests completed per second

**Detailed context:** Devops creates diagnostics scripts that retrieve performance information from multiple layers in the application stack, including infrastructure, to help identify potential bottlenecks during production hours. Devops already understand that the issue is not in the application layers so now they have to dig deeper. Members of devops are typically not experts in infrastructure compute, storage, and networking so they need simple scripts that can provide the information they require without a deep understanding of underlying hardware. With the storage administrator's help, devops use a few simple GETs on storage objects related to their application. To help simplify the query, the storage administrator tags the volumes associated with the application. Devops can use stack wide performance metrics to quickly isolate potential bottlenecks that may be contributing to the slow down in the application.

**Preconditions:**
- Storage system has at least one storage pool with at least one storage volume
- Storage volume is exposed to at least one initiator
- Initiator is driving I/O to the volume

**Feature(s):** Block IO performance

**Inputs:**
- URL of the Swordfish storage service: `/redfish/v1/StorageServices/1`
- Storage pool volume name: 4

**Basic course of events:**

1. uses GET operations to look at metrics of the storage volume

   **Request:** GET `/redfish/v1/StorageServices/1/Volumes/4/Metrics`

   **Headers:** No additional headers required.
   **Body:** None defined.

   **Response:**

   ```json
   {
   "@Redfish.Copyright": "Copyright 2015-2021 SNIA. All rights reserved.",
   "@odata.context": "/redfish/v1/$metadata#VolumeMetrics.VolumeMetrics",
   "@odata.id": "/redfish/v1/StorageServices/1/Volumes/4/Metrics",
   "@odata.type": "#VolumeMetrics.v1_0_0.VolumeMetrics",
   "Id": "Metrics",
   "Name": "Volume 1 Metrics",
   "CurrentPeriod": {
   "BlocksRead": 125534,
   "BlocksWritten": 542653
   },
   "Lifetime": {
   "BlocksRead": 125534,
   "BlocksWritten": 542653
   },
   ```
"PerformanceData": {
  "AverageSecondsPerRead": 1,
  "AverageSecondsPerWrite": 1,
  "AverageSecondsPerTransfer": 1,
  "ReadsPerSecond": 2134,
  "WritesPerSecond": 4325,
  "TransfersPerSecond": 6459,
  "WriteBlocksPerSecond": 3085784,
  "ReadBlocksPerSecond": 9257350,
  "BlocksPerSecond": 12343134
},

"Actions": {
  
  "Volume.ClearCurrentPeriod": {
    "target": "/redfish/v1/StorageServices/1/Volumes/4/Actions/Volume.ClearCurrentPeriod"
  }
},

"Oem": {}

Postconditions: None defined.

See also: None defined.
7.2.19 Make a New Replication Relationship by Creating a Target Consistency Group

**Summary:** Create a new Consistency Group to serve as the target in the replication relationship for an existing source Consistency Group.

**Purpose:** Create a new ConsistencyGroup resource to provide expanded data protection through a replica relationship with the specified source ConsistencyGroup.

**Who:** StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to create a replication relationship for a source ConsistencyGroup when there are no existing ConsistencyGroups that can be assigned as the target.

**Detailed Context:** The admin needs to satisfy a user or application request for a copy of some sort of the original ConsistencyGroup.

**Preconditions:** User has already selected the type of replica, the replica update mode (sync vs async), and the target system and storage pool from which to create the new ConsistencyGroup to serve as the replica.

**Feature(s):** Replication (both local and remote)

**Inputs:**
- URL for Storage Pool: /redfish/v1/Storage/1/StoragePools/PrimaryPool
- Requested replica type: Mirror
- Requested name of ConsistencyGroup (string): CG_DB2
- ReplicaUpdateMode: Synchronous

**Basic Course of Events:**
1. Post (as an Action) the request on the source ConsistencyGroup.

   This instructs the service to use the identified ConsistencyGroup as the source ConsistencyGroup for the specified replication relationship. For any additional details required, the service will rely on default values.

**Request:** POST /redfish/v1/Storage/1/ConsistencyGroups/CG_DB1/ConsistencyGroup.CreateReplicaTarget

**Headers:** No additional headers required.

**Body:**
```
{
   "ConsistencyGroupName" : "CG_DB2",
   "ReplicaUpdateMode" : "Synchronous",
   "TargetStoragePool" : "/redfish/v1/Storage/1/StoragePools/PrimaryPool",
   "ReplicaType" : "Mirror"
}
```

- Response contains the details of the created ConsistencyGroup.

**Response:**

**Headers:** Location = /redfish/v1/Storage/1/ConsistencyGroups/CG_DB1

**Body:**
```
{
   "@SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
   "@odata.context": "/redfish/v1/$metadata#ConsistencyGroup.ConsistencyGroup",
   "@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB1",
   "@odata.type": ":ConsistencyGroup_1_0_0.ConsistencyGroup",
   "Name": "ConsistencyGroup DB1",
   "Id": "1",
   
```
Postconditions: The selected ConsistencyGroup has a new ReplicaTargets property with the link to the new ConsistencyGroup. Elsewhere, there is a new ConsistencyGroup in the system (Name == "CG_DB2", the id set by the system to 23423) that has a ReplicaInfo pointing back to this ConsistencyGroup and which contains all of the replication properties.

Failure Scenario: None defined

See also: Create Replica Target
7.2.20 **Make a New Replication Relationship by Creating a Target Volume**

**Summary:** Create a new volume to serve as a target replica for an existing source volume.

**Purpose:** Create a new volume resource to provide expanded data protection through a replica relationship with the specified source volume.

**Who:** StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to create a replication relationship for a source volume when there are no existing volumes that can be assigned as the target.

**Detailed Context:** The admin needs to satisfy a user or application request for a copy of some sort of the original volume.

**Preconditions:** User has already selected the type of replica, the replica update mode (sync vs async), and the target system and storage pool from which to create the new volume to serve as the replica.

**Feature(s):** Replication (both local and remote)

**Inputs:**
- URL for Storage Pool: `/redfish/v1/Storage/1/StoragePools/PrimaryPool`
- Requested replica type: Mirror
- Requested name of volume (string): Mirror of Volume 65
- ReplicaUpdateMode: Synchronous

**Basic Course of Events:**
1. Post (as an Action) the request on the source Volume.
   
   This instructs the service to use the identified Volume as the source Volume for the specified replication relationship. For any additional details required, the service will rely on default values.

   **Request:** POST `/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1/Volume.CreateReplicaTarget`

   **Headers:** No additional headers required.

   **Body:**

   ```json
   {
   "VolumeName": "Mirror of Volume 65",
   "ReplicaUpdateMode": "Synchronous",
   "TargetStoragePool": "/redfish/v1/Storage/1/StoragePools/PrimaryPool",
   "ReplicaType": "Mirror"
   }
   ```

   - Response contains the details of the created volume.

   **Response:**

   **Headers:** Location = `/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1`

   **Body:**

   ```json
   {
   "@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/2345"
   }
   ```
Postconditions: The selected volume has a new ReplicaTargets property with the link to the new volume. Elsewhere, there is a new volume in the system (Name == “Mirror of Volume 65”, the id set by the system to 2345) that has a ReplicaInfo pointing back to this volume and which contains all of the replication properties.

Failure Scenario: None defined

See also: Create Replica Target (CG)
7.2.21 Provision a Namespace

**Summary:** Provision a Namespace

**Purpose:** Provision a Namespace from a simple NVMe device that does not support Endurance Groups and NVM Sets.

**Who:** StorageAdmin CloudAdmin

**Management Domain:** Block storage management

**Triggers:** None defined.

**Detailed Context:** Create a namespace, and allocate resources reserved for it, from an NVMe device that does not support Endurance Groups and NVM Sets. The Create Namespace request is performed on the Volume Collection in this configuration. The admin needs to satisfy a user or application request to provide a given amount of capacity by creating a new Namespace. Note that this specific use case does not include the assignment of this namespace to an IO Controller.

**Preconditions:** User has already selected the desired capacity.

**Feature(s):** NVMe

**Inputs:**
- URL for Volume Collection: /redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes
- Requested volume size in bytes (1 TiB, here): 1099511627776 (Provisioned Capacity), 1099511627776 (Allocated Capacity)
- Requested name of volume: "MyNamespace"

**Basic Course of Events:**

1. Post the definition of the new Namespace to the Volume resource collection on the NVM Subsystem.

This instructs the service to provision (create) a new Namespace of the requested size. Any additional protection properties will be inherited from the NVM Subsystem.

- **Request:** POST /redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes
  - **Headers:** No additional headers required.
  - **Body:**

```json
{
  "Name": "MyNamespace",
  "Capacity": {
    "Data": {
      "AllocatedBytes": 1099511627776,
      "ProvisionedBytes": 1099511627776
    }
  }
}
```

- **Response:**

Response contains the details of the created volume.

  - **Headers:**
    - Location = /redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyNamespace
  - **Body:**

```json
{
  "@Redfish.Copyright": "Copyright 2014-2021 SNIA. All rights reserved.",
  "@odata.id": "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyNamespace",
  "@odata.type": "#Volume.v1_5_0.Volume",
  "Id": "1",
```
### Postconditions:
The selected Namespace has been created and is added to the NVM Subsystem.

### Failure Scenario:
None defined.

### See also:
NVMe use case to [Attach a Namespace](#), and to [Provision a Namespace with NVMSet](#).
7.2.22 Remove Replication Relationship for a Consistency Group

**Summary:** Disable data synchronization between a source and target Consistency Group, remove the replication relationship, and optionally delete the target Consistency Group.

**Purpose:** The administrator wants to completely delete the relationship between the target and source Consistency Groups.

**Who:** StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to remove a replication relationship due to changing system or environment requirements.

**Detailed Context:** The identified replication relationship is no longer needed, and is to be completely removed from the configuration.

**Preconditions:** User has already identified which replication relationship to delete, and whether or not to retain the target Consistency Group.

**Feature(s):** Replication (both local and remote)

**Inputs:**
- URL for target replica: `/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2`
- Boolean value to determine whether target ConsistencyGroup should be retained or not: `false`

**Basic Course of Events:**

1. Post (as an Action) the request on the source ConsistencyGroup.

This instructs the service to delete the specified replication relationship. For any additional details required, the service will rely on default values.

**Request:** POST `/redfish/v1/Storage/1/ConsistencyGroups/CG_DB1/ConsistencyGroup.RemoveReplicaRelationship`

**Headers:** No additional headers required.

**Body:**

```
{
    "TargetConsistencyGroup": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2",
    "DeleteTargetConsistencyGroup": "false"
}
```

- Response contains the details of the source ConsistencyGroup.

**Response:**

**Headers:** Location = `/redfish/v1/Storage/1/ConsistencyGroups/1`

**Body:**

```
{
    "@SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
    "@odata.context": "/redfish/v1/$metadata#ConsistencyGroup.ConsistencyGroup",
    "@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB1",
    "@odata.type": "#ConsistencyGroup_1_0_0.ConsistencyGroup",
    "Name": "ConsistencyGroup DB1",
    "Id": "1",
    "Description": "Database group 1 primary"
}
```

**Postconditions:** The ConsistencyGroup will no longer have an entry in “ReplicaTargets” for the former replication relationship (the property is not returned in the above example if it were null). Elsewhere, the ConsistencyGroup "CG_DB2" in the system will still exist but will no longer have a StorageReplicaInfo which points back to this ConsistencyGroup.
Failure Scenario: None defined

See also: Remove Replication Relationship
7.2.23 Remove Replication Relationship

Summary: Disable data synchronization between a source and target volume, remove the replication relationship, and optionally delete the target volume.

Purpose: The administrator wants to completely delete the relationship between the target and source volumes.

Who: StorageAdmin, DevOps

Management Domain: Block storage management

Triggers: Need to remove a replication relationship due to changing system or environment requirements.

Detailed Context: The identified replication relationship is no longer needed, and is to be completely removed from the configuration.

Preconditions: User has already identified which replication relationship to delete, and whether or not to retain the target volume.

Feature(s): Replication (both local and remote)

Inputs:
- URL for target replica: /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245
- Boolean value to determine whether target volume should be retained or not: false

Basic Course of Events:

1. Post (as an Action) the request on the source Volume.

   This instructs the service to delete the specified replication relationship. For any additional details required, the service will rely on default values.

Request: POST
   /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1/Volume.RemoveReplicaRelationship

Headers: No additional headers required.

Body:

```
{
    "TargetVolume" : "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245",
    "DeleteTargetVolume" : "false"
}
```

- Response contains the details of the source volume.

Response:

   Headers: Location = /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1

   Body:

```
{
    "SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
    "@odata.context": "/redfish/v1/$metadata#Volume.Volume",
    "@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1",
    "@odata.type": ":Volume_1_4_0.Volume",
    "Name": "Volume 65",
    "Id": "1",
    "Description": ""
}
```

Postconditions: The volume will no longer have an entry in “ReplicaTargets” for the former replication relationship (the property is not returned in the above example if it were null). Elsewhere, the volume “650973452245” in the system will still exist but will no longer have a StorageReplicaInfo which points back to this volume.

Failure Scenario: None defined

See also: Remove Replication Relationship (CG)
7.2.24 Report Namespace Capacity

Summary: Report Namespace Capacity

Purpose: Report Namespace Capacity

Who: StorageAdmin CloudAdmin

Management Domain: Block storage management

Triggers: None defined.

Detailed Context: The namespace capacity information is provided as properties in the namespace object. The NVMe capacity information reported via the Redfish/Swordfish data structures maps the NVMe native capacity information into the RF/SF capacity structures, showing the capacity presented by the namespace to the consumer as AllocatedBytes (e.g., the addressable capacity), and the total available capacity of the namespace as ProvisionedBytes (e.g., the amount of addressable capacity that may actually be used). Note that for a thin provisioned system these values are expected to be different.

Preconditions: None defined.

Feature(s): NVMe

Inputs:
- URL for namespace: /redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/Namespace1

Basic Course of Events:

1. Get the designated Namespace and find the Capacity.Data.AllocatedBytes property within the returned JSON data:

   - **Request**: GET /redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/Namespace1
   - **Headers**: No additional headers required.
   - **Body**:

```json
{
   "@Redfish.Copyright": "Copyright 2014-2021 SNIA. All rights reserved.",
   "$id": "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/Namespace1",
   "$type": "#Volume.v1_5_0.Volume",
   "Id": "1",
   "Name": "Namespace 1",
   "LogicalUnitNumber": 1,
   "Status": {
      "State": "Enabled",
      "Health": "OK"
   },
   "Identifiers": {
      {
         "DurableNameFormat": "NQN",
         "DurableName": "nqn.2014-08.org.nvmexpress:uuid:6c5fe566-10e6-4fb6-aad4-8b4159029384"
      }
   },
   "Capacity": {
      "Data": {
         "ConsumedBytes": 0,
         "AllocatedBytes": 10737418240,
         "ProvisionedBytes": 10737418240
      }
   },
   "CapacitySources": {
      {
         "$id": "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/Namespace1/CapacitySources/Source1"
      }
   }
}
```
• **Response:** None defined.

**Postconditions:** The requested capacity information is returned in the GET request:

```json
{
    "Capacity": {
        "Data": {
            "ConsumedBytes": 0,
            "AllocatedBytes": 10737418240,
            "ProvisionedBytes": 10737418240
        }
    }
}
```

**Failure Scenario:** None defined.

**See also:** None defined.
7.2.25 Report Remaining Life for a Namespace

**Summary**: Report Remaining Life for a Namespace

**Purpose**: Report Remaining Life for a Namespace

**Who**: StorageAdmin CloudAdmin

**Management Domain**: Block storage management

**Triggers**: None defined.

**Detailed Context**: The namespace itself doesn’t have the notion of “remaining life”. Instead, the user should go to the corresponding drive object, and retrieve the “PredictedMediaLifeLeftPercent” property. Note, for a system that has endurance groups and sets, the endurance group also has this property, and the namespace could take the related “PredictedMediaLifeLeftPercent” from its related endurance group as well.

**Preconditions**: The namespace object has a direct link (in the Links property) to its related Drive. This path may not exist in certain types of NVMe devices, such as an Opaque Array.

**Feature(s)**: NVMe (nvme-feature)

**Inputs**:

- URL for Namespace

**Basic Course of Events**:

1. **Request**: GET /redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Volumes/SimpleNamespace

**Headers**: No additional headers required.

**Body**:

```json
{
"@Redfish.Copyright": "Copyright 2014-2021 SNIA. All rights reserved.",
"@odata.id": "/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Volumes/SimpleNamespace",
"@odata.type": "#Volume.v1_5_0.Volume",
"Id": "1",
"Name": "Namespace 1",
"LogicalUnitNumber": 1,
"Status": {
  "State": "Enabled",
  "Health": "OK"
},
"Identifiers": [
  {
    "DurableNameFormat": "NQN",
    "DurableName": "nqn.2014-08.org.nvmexpress:uuid:6c5fe566-10e6-4fb6-aad4-8b4159029384"
  }
],
"Capacity": {
  "Data": {
    "ConsumedBytes": 0,
    "AllocatedBytes": 10737418240
  },
  "Metadata": {
    "AllocatedBytes": 536870912
  }
},
"NVMeNamespaceProperties": {
```
2. GET the Drive related to the Namespace.

* Request: GET /redfish/v1/Chassis/SimplestNVMeSSD/Drives/SimplestNVMeSSD

* Headers: No additional headers required.

* Body:

```json
{
    "@odata.id": "/redfish/v1/Chassis/SimplestNVMeSSD/Drives/SimplestNVMeSSD",
    "@odata.type": "$Drive.v1_9_0.Drive",
    "IndicatorLED": "Lit",
    "Model": "ST9146802SS",
    "Revision": "S20A",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "CapacityBytes": 899527000000,
    "FailurePredicted": false,
    "PredictedMediaLifeLeftPercent": 18,
    "Protocol": "NVMe",
    "MediaType": "SSD",
    "Manufacturer": "Contoso",
    "SerialNumber": "72D0A037FRD26",
    "PartNumber": "SG0GP8811253178M02GJA00",
    "Identifiers": [
        {
            "DurableNameFormat": "NAA",
            "DurableName": "500003942810D13A"
        }
    ],
    "Links": {
        "Volumes": [
            {
                "@odata.id": "/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Volumes/SimpleNamespace"
            }
        ]
    }
}
```
• **Response:** None defined.

**Postconditions:** The drive object returned a “PredictedMediaLifeLeftPercent” value of 18. The user can use this value as the apparent namespace life value.

**Failure Scenario:** None defined.

**See also:** None defined.
7.2.26 Resume the Replication Synchronization Activity for a Consistency Group

**Summary:** Resume the active data synchronization between a source and target Consistency Group, without otherwise altering the replication relationship.

**Purpose:** The administrator wants to restore the relationship between the target and source Consistency Groups, since the temporary condition that led to a suspension of replication has passed.

**Who:** StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to re-activate a suspended replication relationship.

**Detailed Context:** The temporary condition that led to a suspension of replication has passed, and the admin needs to resume replication using the existing replication relationship.

**Preconditions:** User has already identified which target ConsistencyGroup to resume, and implementation preserves replication information when a relationship is suspended.

**Feature(s):** Replication (both local and remote)

**Inputs:**
- URL for target replica: `/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2`

**Basic Course of Events:**

1. Post (as an Action) the request on the source ConsistencyGroup.

This instructs the service to use the identified ConsistencyGroup as the source ConsistencyGroup for the specified replication relationship. For any additional details required, the service will rely on default values.

**Request:** POST /redfish/v1/Storage/1/ConsistencyGroups/DB_CG1/ConsistencyGroup.ResumeReplication

**Headers:** No additional headers required.

**Body:**

```
{
  "TargetConsistencyGroup": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2"
}
```

- Response contains the details of the source ConsistencyGroup.

**Response:**

**Headers:** Location = /redfish/v1/Storage/1/ConsistencyGroups/CG_DB1

**Body:**

```
{
  "@SSM.Copyright": "Copyright © 2014-2021 SNIA. All rights reserved.",
  "@odata.context": "/redfish/v1/$metadata#ConsistencyGroup.ConsistencyGroup",
  "@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB1",
  "@odata.type": "<<ConsistencyGroup_1_0_0.ConsistencyGroup",
  "Name": "ConsistencyGroup DB1",
  "Id": "1",
  "Description": "Database group 1 primary",
  "ReplicaTargets": ["@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2"]
}
```
**Postconditions:** The selected ConsistencyGroup has an updated ReplicaTargets entry for the new relationship. Elsewhere, there is a ConsistencyGroup “CG_DB2” in the system that has a ReplicaInfo which points back to this ConsistencyGroup and which contains all of the Replica configuration information. The replica state in the target ConsistencyGroup has been updated according to the requested action (e.g., from “suspended” to “active”).

**Failure Scenario:** None defined

See also: Resume the Replication Synchronization Activity
7.2.27 Resume the Replication Synchronization Activity

**Summary:** Resume the active data synchronization between a source and target volume, without otherwise altering the replication relationship.

**Purpose:** The administrator wants to restore the relationship between the target and source volumes, since the temporary condition that led to a suspension of replication has passed.

**Who:** StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to re-activate a suspended replication relationship.

**Detailed Context:** The temporary condition that led to a suspension of replication has passed, and the admin needs to resume replication using the existing replication relationship.

**Preconditions:** User has already identified which target volume to resume, and implementation preserves replication information what a relationship is suspended.

**Feature(s):** Replication (both local and remote)

**Inputs:**
- URL for target replica: `/redfish/v1/Storage/1/Volumes/650973452245`

**Basic Course of Events:**

1. Post (as an Action) the request on the source Volume.

This instructs the service to use the identified Volume as the source Volume for the specified replication relationship. For any additional details required, the service will rely on default values.

**Request:** POST `/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1/Volume.ResumeReplication`

**Headers:** No additional headers required.

**Body:**

```json
{
    "TargetVolume": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245"
}
```

- Response contains the details of the source volume.

**Response:**

**Headers:** Location = `/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1`

**Body:**

```json
{
    "@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245",
    "Name": "Volume 65",
    "Id": "1",
    "Description": "",
    "ReplicaTargets": [{"@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245"}]
}
```

**Postconditions:** The selected volume has an updated ReplicaTargets entry for the new relationship. Elsewhere, there is a volume “650973452245” in the system that has a ReplicaInfo which points back to this volume and which contains all of the Replica configuration information. The replica state in the target volume has been updated according to the requested action (e.g., from “suspended” to “active”).
**Failure Scenario:** None defined

See also: Resume the Replication Synchronization Activity (CG)
### 7.2.28 Reverse a Replication Relationship for Consistency Groups

**Summary:** Reverse the replication relationship between a source and target Consistency Group.

**Purpose:** The administrator wants to reconfigure the relationship between the target and source Consistency Groups, reversing their roles.

**Who:** StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to reconfigure existing storage due to changing system or environment requirements.

**Detailed Context:** The admin needs to satisfy a user or application request swapping the target/source roles in a replication relationship.

**Preconditions:** User has already identified which target ConsistencyGroup and replication relationship to reverse.

**Feature(s):** Replication (both local and remote)

**Inputs:**
- URL for target replica: `/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2`

**Basic Course of Events:**

1. Post (as an Action) the request on the source ConsistencyGroup.

This instructs the service to use the identified ConsistencyGroup as the source ConsistencyGroup for the specified replication relationship. For any additional details required, the service will rely on default values.

**Request:** POST /redfish/v1/Storage/1/ConsistencyGroups/DB_CG1/ConsistencyGroup.ReverseReplicationRelationship

**Headers:** No additional headers required.

**Body:**

```
{
  "TargetConsistencyGroup": "/redfish/v1/Storage/1/ConsistencyGroups/DB_CG2"
}
```

- Response contains the details of the source ConsistencyGroup.

**Response:**

**Headers:** Location = /redfish/v1/Storage/1/ConsistencyGroups/1

**Body:**

```
{
  "@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB1",
  "Name": "Consistency Group DB1",
  "Id": 1,
  "Description": "Database group 1 primary",
  "ReplicaInfos": [{
    "ReplicaState": "Failedover",
    "ReplicaType": "Snapshot",
    "ReplicaPriority": "Default",
    "ReplicaRecoveryMode": "Manual",
    "ReplicaUpdateMode": "Asynchronous",
    "Replica": {
      "@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/DB_CG2"
    }
  }]
}
```
**Postconditions:** The selected ConsistencyGroup will now have an updated ReplicaInfo for the relationship, which contains the replication attributes and a pointer to the source replica. Elsewhere, there is a ConsistencyGroup “DB_CG2” in the system that now has an ReplicaTargets entry that points back to this ConsistencyGroup ("@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/DB_CG1").

**Failure Scenario:** None defined

See also: Reverse a Replication Relationship
7.2.29 Reverse a Replication Relationship

Summary: Reverse the replication relationship between a source and target volume.

Purpose: The administrator wants to reconfigure the relationship between the target and source volumes, reversing their roles.

Who: StorageAdmin, DevOps

Management Domain: Block storage management

Triggers: Need to reconfigure existing storage due to changing system or environment requirements.

Detailed Context: The admin needs to satisfy a user or application request swapping the target/source roles in a replication relationship.

Preconditions: User has already identified which target volume and replication relationship to reverse.

Feature(s): Replication (both local and remote)

Inputs:
- URL for target replica: /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245

Basic Course of Events:
1. Post (as an Action) the request on the source Volume.

This instructs the service to use the identified Volume as the source Volume for the specified replication relationship. For any additional details required, the service will rely on default values.

Request: POST
/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1/Volume.ReverseReplicationRelationship

Headers: No additional headers required.

Body:

```
{
   "TargetVolume": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245"
}
```

- Response contains the details of the source volume.

Response:

Headers: Location = /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1

Body:

```
{
   "@SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
   "@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1",
   "@odata.type": "#Volume_1_6_0.Volume",
   "Name": "Volume 65",
   "Id": "1",
   "Description": "",
   "ReplicaInfos": [{
      "ReplicaState": "Failedover",
      "ReplicaType": "Snapshot",
      "ReplicaPriority": "Default",
      "ReplicaRecoveryMode": "Manual",
      "ReplicaUpdateMode": "Asynchronous",
      "Replica": {"@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245"}
   }]
}
```
**Postconditions:** The selected volume will now have an updated ReplicaInfo for the relationship, which contains the replication attributes and a pointer to the source replica. Elsewhere, there is a volume “650973452245” in the system that now has an ReplicaTargets entry that points back to this volume ("@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1").

**Failure Scenario:** None defined

See also: Reverse a Replication Relationship (CG)
7.2.30 Split a Replica

Summary: Split the replication relationship and suspend data synchronization between a source and target volume.

Purpose: The administrator wants to reconfigure the relationship between the target and source volumes.

Who: StorageAdmin, DevOps

Management Domain: Block storage management

Triggers: Need to reconfigure existing storage due to changing system or environment requirements.

Detailed Context: The admin needs to satisfy a user or application request to change the existing configuration between the target and source volumes in a replication relationship.

Preconditions: User has already identified which target volume and replication relationship to split.

Feature(s): Replication (both local and remote)

Inputs:
- URL for target replica: /redfish/v1/Storage/1/Volumes/650973452245

Basic Course of Events:

1. Post (as an Action) the request on the source Volume.

This instructs the service to use the identified Volume as the source Volume for the specified replication relationship. For any additional details required, the service will rely on default values.

Request: POST /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1/Volume.SplitReplication

Headers: No additional headers required.

Body:

```json
{
  "TargetVolume": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245"
}
```

- Response contains the details of the source volume.

Response:

Headers: Location = /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1

Body:

```json
{
  "$SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
  "$@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1",
  "$@odata.type": "$Volume_1_6_0.Volume",
  "Name": "Volume 65",
  "Id": "1",
  "Description": "",
  "ReplicaTargets": [{"$@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245"}]
}
```

Postconditions: The selected volume has a new ReplicaTargets property with the link to the volume. Elsewhere, there is a volume “650973452245” in the system that has a ReplicaInfo pointing back to this volume and which contains all of the Replica configuration information. The replica state in the target volume has been updated according to the requested action (e.g., from “active” to “suspended / split”).

Failure Scenario: None defined

See also: Split a Replica (CG)
### 7.2.31 Split a set of Replicas in Consistency Groups

**Summary:** Split the replication relationship and suspend data synchronization between a source and target ConsistencyGroup.

**Purpose:** The administrator wants to reconfigure the relationship between the target and source ConsistencyGroups.

**Who:** StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to reconfigure existing storage due to changing system or environment requirements.

**Detailed Context:** The admin needs to satisfy a user or application request to change the existing configuration between the target and source ConsistencyGroups in a replication relationship.

**Preconditions:** User has already identified which target ConsistencyGroup and replication relationship to split.

**Feature(s):** Replication (both local and remote)

**Inputs:**
- URL for target replica: `/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2`

**Basic Course of Events:**

1. Post (as an Action) the request on the source ConsistencyGroup.

This instructs the service to use the identified ConsistencyGroup as the source ConsistencyGroup for the specified replication relationship. For any additional details required, the service will rely on default values.

**Request:** POST /redfish/v1/Storage/1/ConsistencyGroups/CG_DB1/ConsistencyGroup.SplitReplication

**Headers:** No additional headers required.

**Body:**

```json
{
  "TargetConsistencyGroup": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2"
}
```

- Response contains the details of the source ConsistencyGroup.

**Response:**

**Headers:** Location = /redfish/v1/Storage/1/ConsistencyGroups/1

**Body:**

```json
{
  "@SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
  "@odata.context": "/redfish/v1/$metadata#ConsistencyGroup.ConsistencyGroup",
  "@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB1",
  "@odata.type": "#ConsistencyGroup_1_0_0.ConsistencyGroup",
  "Id": "1",
  "Name": "ConsistencyGroup DB1",
  "Description": "Database group 1 primary",
  "ReplicaTargets": 
  [
    
    {"@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2"}
  ]
}
```

**Postconditions:** The selected ConsistencyGroup has a new ReplicaTargets property with the link to the ConsistencyGroup. Elsewhere, there is a ConsistencyGroup "650973452245" in the system that has a ReplicaInfo pointing back to this ConsistencyGroup and which contains all of the Replica configuration information. The replica state in the target ConsistencyGroup has been updated according to the requested action (e.g., from "active" to "suspended / split").
Failure Scenario: None defined

See also: Split a Replica
7.2.32 Subscribe to Threshold Events

Summary: Subscribe to Trigger/Clear events for LowSpaceWarningThresholds for a named Volume.

Purpose: Provide an event stream to support utilization management. This is used in conjunction with LowSpaceWarningThresholds to provide a means for on-going monitoring of resource consumption.

Who: CloudAdmin, StorageAdmin, DevOps

Management domain: Block storage management, Application storage management

Triggers: None defined.

Detailed Context: This provides the basis for monitoring capacity consumption.

Preconditions: None defined.

Feature(s): Event notification

Inputs:
- The URL of the StorageService: /redfish/v1/StorageServices/Simple1
- The Volume to be monitored: Vol1
- The subscription destination: "http://www.dnsname.com/Destination1"
- An array of events to be subscribed:

  ```json
  ["LowSpaceWarningThresholdTriggered", "LowSpaceWarningThresholdCleared"]
  ```

Basic course of events:

1. Retrieve the event destination

   * Request: POST /redfish/v1/EventService/Subscriptions/Members
     - Headers: No additional headers required.
     - Body:

     ```json
     {
       "@Redfish.Copyright": "Copyright 2016-2021 Storage Networking Industry Association (SNIA), USA. All rights reserved. For the full SNIA copyright policy, see http://www.snia.org/about/corporate_info/copyright",
       "@odata.context": "/redfish/v1/$metadata#EventDestination.EventDestination",
       "@odata.type": "#EventDestination.v1_0_2.EventDestination",
       "Name": "Volume1 Usage Threshold",
       "Destination": "http://www.dnsname.com/Destination1",
       "EventTypes": [
         "Alert"
       ],
       "Context": "WebUser3",
       "Protocol": "Redfish",
       "OriginResources": [{"@odata.id": "/redfish/v1/StorageServices/Simple1/Volumes/Vol1"}],
       "MessageIds": [{"LowSpaceWarningThresholdTriggered", "LowSpaceWarningThresholdCleared"}]
     }
     ```

   - Response: 201 - Created

     ```json
     {
       "Location": "/redfish/v1/EventService/Subscriptions/1/Members/1e7da"
     }
     ```

   Postconditions: Newly-created event subscription is added to the EventService.

Failure Scenario: None defined.
See also: None defined.
7.2.33 Suspend Replication Synchronization Activity between Consistency Groups

**Summary:** Suspend active data synchronization between a source and target ConsistencyGroup, without otherwise altering the replication relationship.

**Purpose:** Due to temporarily changed environmental constraints, the administrator wants to change the level of data protection between the target and source ConsistencyGroups.

**Who:** StorageAdmin, DevOps

**Management Domain:** Block storage management

**Triggers:** Need to reconfigure existing storage due to changing system or environment requirements.

**Detailed Context:** The admin needs to satisfy a user or application request to change the existing configuration between the existing target and source ConsistencyGroups in a replication relationship, without deleting the relationship.

**Preconditions:** User has already identified which target ConsistencyGroup to suspend.

**Feature(s):** Replication (both local and remote)

**Inputs:**
- URL for target replica: `/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2`

**Basic Course of Events:**
1. Post (as an Action) the request on the source ConsistencyGroup.
   
   This instructs the service to use the identified ConsistencyGroup as the source ConsistencyGroup for the specified replication relationship. For any additional details required, the service will rely on default values.

**Request:** POST /redfish/v1/Storage/1/ConsistencyGroups/CG_DB1/ConsistencyGroup.SuspendReplication

**Headers:** No additional headers required.

**Body:**
```json
{
   "TargetConsistencyGroup": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2"
}
```

- Response contains the details of the source ConsistencyGroup.

**Response:**

**Headers:** Location = /redfish/v1/Storage/1/ConsistencyGroups/CG_DB1

**Body:**
```json
{
   "@SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
   "@odata.context": "/redfish/v1/$metadata#ConsistencyGroup.ConsistencyGroup",
   "@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB1",
   "@odata.type": "&ConsistencyGroup_1_0_0.ConsistencyGroup",
   "Name": "ConsistencyGroup DB1",
   "Id": "1",
   "Description": "Database group 1 primary",
   "ReplicaTargets": [{"@odata.id": "/redfish/v1/Storage/1/ConsistencyGroups/CG_DB2"}]
}
```

**Postconditions:** The selected ConsistencyGroup has an updated ReplicaTargets entry for the new relationship. Elsewhere, there is a ConsistencyGroup “CG_DB2” in the system that has a ReplicaInfo which points back to this ConsistencyGroup and which contains all of the Replica configuration information. The replica state in the target ConsistencyGroup has been updated according to the requested action (e.g., from "Active" to “Suspended”).

**Failure Scenario:** None defined
See also: Suspend Replication Synchronization Activity
7.2.34 Suspend Replication Synchronization Activity

Summary: Suspend active data synchronization between a source and target volume, without otherwise altering the replication relationship.

Purpose: Due to temporarily changed environmental constraints, the administrator wants to change the level of data protection between the target and source volumes.

Who: StorageAdmin, DevOps

Management Domain: Block storage management

Triggers: Need to reconfigure existing storage due to changing system or environment requirements.

Detailed Context: The admin needs to satisfy a user or application request to change the existing configuration between the existing target and source volumes in a replication relationship, without deleting the relationship.

Preconditions: User has already identified which target volume to suspend.

Feature(s): Replication (both local and remote)

Inputs:
- URL for target replica: /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245

Basic Course of Events:

1. Post (as an Action) the request on the source Volume.

This instructs the service to use the identified Volume as the source Volume for the specified replication relationship. For any additional details required, the service will rely on default values.

Request: POST /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1/Volume.SuspendReplication

Headers: No additional headers required.

Body:

```
{
    "TargetVolume": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245"
}
```

- Response contains the details of the source volume.

Response:

Headers: Location = /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1

Body:

```
{
    "@SSM.Copyright": "Copyright (c) 2014-2021 SNIA. All rights reserved.",
    "@odata.context": "/redfish/v1/$metadata#Volume.Volume",
    "@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1",
    "@odata.type": "+Volume_1_4_0.Volume",
    "Name": "Volume 65",
    "Id": "1",
    "Description": "",
    "ReplicaTargets": [
        {
            "@odata.id": "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650973452245"
        }
    ]
}
```

Postconditions: The selected volume has an updated ReplicaTargets entry for the new relationship. Elsewhere, there is a volume "650973452245" in the system that has a ReplicaInfo which points back to this volume and which contains all of the Replica configuration information. The replica state in the target volume has been updated according to the requested action (e.g., from "Active" to "Suspended").

Failure Scenario: None defined
See also: Suspend Replication Synchronization Activity (CG)