



Swordfish Scalable Storage Management API Users Guide

Version: 1.2.6

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.

SNIA Approved Publication

This document has been released and approved by the SNIA. The SNIA believes that the ideas, methodologies, and technologies described in this document accurately represent the SNIA goals and are appropriate for widespread distribution. Suggestion for revision should be directed to <http://www.snia.org/feedback/>.

Last Updated: 9 April 2024

Contents

USAGE	7
DISCLAIMER	8
Current Revision	8
Contact SNIA	8
FEEDBACK AND INTERPRETATIONS	9
INTENDED AUDIENCE	9
VERSIONING POLICY	9
Revision History	10
About SNIA	14
Acknowledgements	14
1 Introduction	16
1.1 Audience	16
1.2 Documentation structure	16
1.3 Implementation scope assumptions	17
1.4 Base implementation assumptions	18
1.5 Knowledge assumptions	18
1.6 Related documents	18
2 General query syntax	19
2.1 Query method	19
2.2 Query Headers	19
2.2.1 Request headers	19
2.2.2 Response headers	21
2.3 Service root	23
2.4 Resource path	23
2.5 Query options	24
2.6 Filter expressions	25
2.7 HTTP status codes	25
3 User Guidance	26
4 Features	27
4.1 Overview	27

4.2	Access management feature	27
4.2.1	Overview	27
4.2.2	Feature-specific Guidance	28
4.2.3	Use Cases	28
4.3	Block provisioning feature	28
4.3.1	Overview	28
4.3.2	Feature-specific Guidance	28
4.3.3	Use Cases	28
4.4	Capacity management feature	29
4.4.1	Overview	29
4.4.2	Feature-specific Guidance	29
4.4.3	Use Cases	29
4.5	Class of Service Features	29
4.5.1	Overview	29
4.5.2	Feature-specific Guidance	29
4.5.3	Use Cases	30
4.6	Event notification feature	30
4.6.1	Overview	30
4.6.2	Feature-specific Guidance	30
4.6.3	Use Cases	30
4.7	File provisioning feature	30
4.7.1	Overview	30
4.7.2	Feature-specific Guidance	30
4.7.3	Use Cases	31
4.8	Block IO performance feature	31
4.8.1	Overview	31
4.8.2	Feature-specific Guidance	31
4.8.3	Use Cases	31
4.9	Block mapping and masking feature	31
4.9.1	Overview	31
4.9.2	Feature-specific Guidance	31
4.9.3	Use Cases	31
4.10	NVMe Support feature	31
4.10.1	Overview	31
4.10.2	Feature-specific Guidance	32

4.10.3	Use Cases	32
4.11	Replication Feature	32
4.11.1	Overview	32
4.11.2	Feature-specific Guidance	32
4.11.3	Use Cases	33
5	Use cases	34
5.1	Alphabetic list of use cases	34
5.1.1	<i>Add Multiple Drives to an Existing Storage Pool</i>	34
5.1.2	<i>Attach a Namespace</i>	36
5.1.3	<i>Can a new Namespace be created?</i>	39
5.1.4	<i>Can a new Namespace be created?</i>	40
5.1.5	<i>Can a new Volume be created?</i>	41
5.1.6	<i>Change only the RAID Type of an Existing Volume</i>	42
5.1.7	<i>Change only the span count of an existing volume</i>	49
5.1.8	<i>Confirm valid LBA formats</i>	56
5.1.9	<i>Create a new connection to an existing volume</i>	60
5.1.10	<i>Create a new endpoint</i>	64
5.1.11	<i>Create a new endpoint group</i>	67
5.1.12	<i>Create a New Replication Relationship by Assigning an existing Tar- get Consistency Group</i>	71
5.1.13	<i>Create a New Replication Relationship by Assigning a Target Volume</i>	73
5.1.14	<i>Create an on-demand snapshot of a Volume</i>	75
5.1.15	<i>Create class of service</i>	79
5.1.16	<i>Create ConsistencyGroup</i>	81
5.1.17	<i>Create file share</i>	84
5.1.18	<i>Create file system</i>	87
5.1.19	<i>Create file system</i>	89
5.1.20	<i>Create line of service</i>	93
5.1.21	<i>Create storage pool and specify a pool type</i>	96
5.1.22	<i>Create storage pool</i>	99
5.1.23	<i>Create storage pool using Specified Set of Drives and RAIDTypes</i>	101
5.1.24	<i>Create storage pool using specified set of drives</i>	105
5.1.25	<i>Create Volume specifying Class of Service</i>	110
5.1.26	<i>Create Volume using Default Class of Service</i>	114
5.1.27	<i>Delete an endpoint</i>	118

5.1.28	<i>Delete Multiple Drives from an Existing Storage Pool</i>	120
5.1.29	<i>Delete Volume</i>	122
5.1.30	<i>Deprovision a Namespace</i>	124
5.1.31	<i>Detach a Namespace</i>	125
5.1.32	<i>Expand capacity of a storage volume</i>	130
5.1.33	<i>Make a New Replication Relationship by Creating a Target Consistency Group</i>	137
5.1.34	<i>Make a New Replication Relationship by Creating a Target Volume</i>	139
5.1.35	<i>Provision a Namespace from NVM Set</i>	141
5.1.36	<i>Provision a Namespace</i>	145
5.1.37	<i>Provision a Namespace with a specific LBA format</i>	149
5.1.38	<i>Query Supported Security Protocols</i>	152
5.1.39	<i>Receive Security Protocol Data</i>	154
5.1.40	<i>Remove Replication Relationship for a Consistency Group</i>	156
5.1.41	<i>Remove Replication Relationship</i>	158
5.1.42	<i>Report Namespace Capacity</i>	160
5.1.43	<i>Report Remaining Life for a Namespace</i>	163
5.1.44	<i>Resume the Replication Synchronization Activity for a Consistency Group</i>	168
5.1.45	<i>Resume the Replication Synchronization Activity</i>	170
5.1.46	<i>Retrieve latest instance of storage metrics information</i>	172
5.1.47	<i>Reverse a Replication Relationship for Consistency Groups</i>	174
5.1.48	<i>Reverse a Replication Relationship</i>	176
5.1.49	<i>Review Metrics Trends</i>	178
5.1.50	<i>Send Security Protocol Data</i>	180
5.1.51	<i>Split a Replica</i>	183
5.1.52	<i>Split a set of Replicas in Consistency Groups</i>	185
5.1.53	<i>Subscribe to Threshold Events</i>	187
5.1.54	<i>Suspend Replication Synchronization Activity between Consistency Groups</i>	189
5.1.55	<i>Suspend Replication Synchronization Activity</i>	192
5.1.56	<i>Update access rights on an existing volume</i>	194
5.1.57	<i>Use Features Registry to confirm functionality</i>	200

List of Tables

1	Revision history	10
2	Contributors	14
3	Guidelines for the Use Case Template	17
4	Query methods	19
5	Request headers	20
6	Response headers	21
7	Query Options	24

USAGE

Copyright (c) 2016 - 2024 Storage Networking Industry Association. All rights reserved. All other trademarks or registered trademarks are the property of their respective owners.

Storage Networking Industry Association (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 SNIA copyright on that material, and must credit 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

Copyright (c) 2024, Storage Networking Industry Association.

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 Storage Networking Industry Association 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 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. 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/>.

Current Revision

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

Contact SNIA

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 SNIA, 5201 Great America Parkway, Suite 320, Santa Clara, CA 95054, USA.

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 1: Revision history

Date	Rev	Notes
19 September 2016	1.0.0	Initial Release
12 October 2016	1.0.1	General clean up and formatting consistency A discussion of unused CoS and LoS entries in ServiceCatalog Improve purpose for many use cases
1 November 2016	1.0.2	Corrected XREF link formatting
24 January 2017	1.0.3	Additional use cases and new document section addressing client considerations
25 April 2017	1.0.4	Update cross-references
3 October 2017	1.0.5	Minor updates and corrections
13 February 2018	1.0.6	Added on-demand replication use cases
12 October 2018	1.0.7	Editorial cleanup of JSON
22 August 2019	1.1.0	Restructured to add features and feature cross references, and many new use cases added: Create Volume for multiple scenarios (including Redfish Storage)

Date	Rev	Notes
		Create Storage Pool for multiple scenarios
		Replication use cases using single Volume
		Replication use cases using Consistency Groups
12 November 2019	1.1.0	Released as Technical Position
12 November 2019	1.1.0a	Released as Corrected Technical Position
		Formatting fixes – word wrap in pdf doc format to fix truncated lines
		Added cross referencing of Features to use cases
		Editorial changes and cleanup
18 August 2020	1.2.1	Added NVMe-specific use cases
31 October 2020	1.2.1c	Released as SNIA Approved Publication
2 March 2021	1.2.2	Added cross-references to NVMe mapping document
		Added new use cases for StoragePool actions.
		Errata fixes.
14 June 2021	1.2.2a	Released as SNIA Approved Publication
30 August 2021	1.2.3	Update references in examples
		Add use cases for ChangeRAIDType, ChangeSpanCount.
5 December 2021	1.2.3	Release as SNIA Approved Publication

Date	Rev	Notes
12 April 2022	1.2.4	Release as Working Draft. Add multiple new use cases for access management and volume creation through capabilities. Enhance language and examples responses throughout. Add use of content-type headers.
12 July 2022	1.2.4a	Release as SNIA Standard.
16 March 2023	1.2.5	Add Create Consistency Group use case Add new section for security related use cases. New use cases include: Query Supported Security Protocols SecuritySend Usage (NVMe example) SecurityReceive Usage (NVMe example) Add Etag headers to use case responses Errata Fixes: Correct return value in Create New Connection use case
20 June 2023	1.2.5a	Release as a SNIA Standard. Remove erroneous use case entry.
22 January 2024	1.2.6	Release as Working Draft. Copyrights updated to 2024. Updates SNIA Front Matter.

Date	Rev	Notes
		<p>Simplified and updated Management Domains, Actors.</p> <p>Add new use cases (access management):</p> <ul style="list-style-type: none">• Create Endpoint• Delete Endpoint• Create Endpoint Group <p>Add new use cases (block provisioning):</p> <ul style="list-style-type: none">• Delete Volume from Storage Pool <p>Add new use cases (capacity management):</p> <ul style="list-style-type: none">• Review Metric Trends <p>Add new use cases (NVMe):</p> <ul style="list-style-type: none">• Namespace Creation Permission <p>Updated use cases (IOPerformance):</p> <ul style="list-style-type: none">• Retrieve Volume Metrics <p>Updated use cases (Security Management):</p> <ul style="list-style-type: none">• Security Send• Security Receive <p>Errata corrections to use cases:</p> <ul style="list-style-type: none">• Create New Connection, Create Consistency Group, Create Volume from Storage Pool.
9 April 2024	1.2.6	Release as SNIA Standard

About SNIA

SNIA is a not-for-profit global organization made up of corporations, universities, startups, and individuals. The members collaborate to develop and promote vendor-neutral architectures, standards, and education for management, movement, and security for technologies related to handling and optimizing data. SNIA focuses on the transport, storage, acceleration, format, protection, and optimization of infrastructure for data. Learn more at www.snia.org.

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

Member	Representatives (* – prior employer)
Broadcom Inc.	Richelle Ahlvers *
Celestica	Krishnakumar Gowravaram
Cisco Systems, Inc.	Krishnakumar Gowravaram *
Code Construct	Jeremy Kerr
Dell Inc.	Patrick Boyd
	George Ericson
	Jim Pendergraft
	Sean McGinnis
	Michael Raineri
	Rich Roscoe
Futurewei Inc.	Sean McGinnis *
Hitachi Data Systems	Eric Hibbard
Hewlett Packard Enterprise	Curtis Ballard

Member	Representatives (* – prior employer)
Inova Development Inc.	Jeff Hilland
	Chris Lionetti
	John Mendonca
	Doug Voigt
Intel Corporation	Karl Schopmeyer
	Richelle Ahlvers
	Rajalaxmi Angadi
	Klaudia Jablonska
	Phil Cayton
	Mariusz Krzywienski
	Slawek Putyrski
	Paul von Behren
Microsemi Corporation	Anand Nagarjan
Microsoft Corporation	Hector Linares
	Jim Pinkerton
	Michael Pizzo
	Scott Seligman
NetApp, Inc.	Don Deel
	Fred Knight
	Nilesh Maheshwari
ScienceLogic	Patrick Strick
VMware, Inc.	Murali Rajagopal

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 3: Guidelines for the Use Case Template

Name	Description
Title	A description of the high-level scope of the Use Case
Summary	A high-level summary of the use case
Purpose	The intended goals or motivations for the use case
Triggers	A description of likely business conditions or goals that would make this use case useful
Detailed Context	A detailed description of the operations environment and configuration assumptions for this use case
Preconditions	Pre-existing knowledge, configurations or capabilities
Inputs	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))
Basic Course of Events	A sequence of API requests, including required headers, the body of the request, and the expected reply
Configuration Impacts	Changes to the storage configurations caused by the use case
Failure Scenario	Common failure conditions encountered in this use case
See Also	Other Use Cases that may be of interest

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.

In the same vein, the example query responses are not intended to perfectly reproduce what would be returned from any given implementation. Rather, they focus on illustrating a particular functionality or approach, and include only the information that is essential to that end.

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](#). 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 RESTful APIs: [Wikipedia](#)
- For Redfish standards and related material: [Redfish home page](#)
- For HTTP standards: [Wikipedia](#)

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](#)
- [Swordfish Tutorials](#)
- [Swordfish NMVe Mapping Guide](#)
- [Swordfish Profile Bundle](#)
- [Redfish Specification](#)

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 4: Query methods

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

2.2 Query Headers

All HTTP requests and responses in a compliant Swordfish implementation support the HTTP headers required by the [Redfish Protocol Specification](#). 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 5: Request headers

Header	Support ed Values	Notes
Accept	RFC 7231	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.
Content-Type	RFC 7231	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.
OData-Version	4.0	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.
Authori- zation	RFC 7235 , Section 4.2	Required for Basic Authentication
User-Ag- ent	RFC 7231	Required for tracing product tokens and their version. Multiple product tokens may be listed.
Host	RFC 7230	Required to allow support of multiple origin hosts at a single IP address.
Origin	W3C CORS , Section 5.7	Used to allow web applications to consume Redfish Service while preventing CSRF attacks.

Header	Support ed Values	Notes
If-Matc h	RFC 7232	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.
X-Auth-Token	Opaque encoded octet strings	Used for authentication of user sessions. The token value shall be indistinguishable from random.

2.2.2 Response headers

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

Table 6: Response headers

Header	Support ed Values	Notes
OData-V ersion	4.0	Describes the OData version of the payload that the response conforms to.
Content -Type	RFC 7231	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.
ETag	RFC 7232	An identifier for a specific version of a resource, often a message digest. Etags shall be included on responses to GETs of ManagerAccount objects.

Header	Support ed	
	Values	Notes
Server	RFC 7231	Required to describe a product token and its version. Multiple product tokens may be listed.
Link		Link headers shall be returned as described in the clause on Link Headers
Locatio n	RFC 7231	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.
Cache-C ontrol	RFC 7234	This header shall be supported and is meant to indicate whether a response can be cached or not.
Access- Control -Allow- Origin	W3C CORS , Section 5.7	Prevents or allows requests based on originating domain. Used to prevent CSRF attacks.
Allow	POST, PUT, PATCH, DELETE, GET, HEAD	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.
WWW-Aut henticat e	RFC 7234 , Section 4.1	Required for Basic and other optional authentication mechanisms. See the Security clause for details.
X-Auth- Token	Opaque encoded octet strings	Used for authentication of user sessions. The token value shall be indistinguishable from random.

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 7: Query Options

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

2.6 Filter expressions

Simple example:

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

For more information see [Filter Expression](#) 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](#) and the [Redfish Specification](#).

3 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.

4 Features

4.1 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 grouped 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:

- [Access management](#)
- [Block capacity management](#)
- [Block IO performance](#)
- [Block provisioning](#)
- [Connectivity Management](#)
- Discovery
- EnergyStar
- [Event notification](#)
- [File provisioning](#)
- File capacity management
- [Mapping and masking](#)
- NVMe
- [Replication \(both local and remote\)](#)
- [Security Management](#)

4.2 Access management feature

4.2.1 Overview

This feature provides access management configuration and control.

4.2.2 Feature-specific Guidance

None defined.

4.2.3 Use Cases

- [Create a new endpoint](#)
- [Delete an endpoint](#)
- [Create a new endpoint group](#)
- [Create a new connection to an existing volume](#)
- [Create ConsistencyGroup](#)

4.3 Block provisioning feature

4.3.1 Overview

This feature provides basic, block-based storage provisioning.

4.3.2 Feature-specific Guidance

None defined.

4.3.3 Use Cases

- [Add Multiple Drives to an Existing Storage Pool](#)
- [Change only the RAID Type of an Existing Volume](#)
- [Change only the span count of an existing volume](#)
- [Create Volume from an Existing Storage Pool](#)
- [Delete Multiple Drives from an Existing Storage Pool](#)
- [Delete Volume](#)
- [Can a new Namespace be created?](#)
- [Can a new Volume be created?](#)

4.4 Capacity management feature

4.4.1 Overview

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

4.4.2 Feature-specific Guidance

None defined.

4.4.3 Use Cases

- [Create storage pool using specified set of drives](#)
- [Create storage pool using Specified Set of Drives and RAIDTypes](#)
- [Create storage pool and specify a pool type](#)
- [Expand capacity of a storage volume](#)
- [Review Metrics Trends](#)

4.5 Class of Service Features

4.5.1 Overview

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

4.5.2 Feature-specific Guidance

4.5.2.1 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.

4.5.3 Use Cases

- [Create class of service](#)
- [Create file system](#)
- [Create line of service](#)
- [Create storage pool](#)
- [Create Volume specifying Class of Service](#)
- [Create Volume using Default Class of Service](#)
- [Create an on-demand snapshot of a Volume](#)

4.6 Event notification feature

4.6.1 Overview

This feature provides basic eventing.

4.6.2 Feature-specific Guidance

The use cases defined here are limited to event functionality required for storage management with Swordfish, and do not attempt to cover the full breadth of eventing support provided by Redfish. That information can be found in the [Redfish specification](#).

4.6.3 Use Cases

- [Subscribe to Threshold Events](#)

4.7 File provisioning feature

4.7.1 Overview

This feature provides file-based storage provisioning.

4.7.2 Feature-specific Guidance

None defined.

4.7.3 Use Cases

- [Create file share](#)
- [Create file system](#)

4.8 Block IO performance feature

4.8.1 Overview

This feature provides basic performance metrics.

4.8.2 Feature-specific Guidance

None defined.

4.8.3 Use Cases

- [Retrieve latest instance of storage metrics information](#)

4.9 Block mapping and masking feature

4.9.1 Overview

This feature provides block-based mapping and assignment of storage volumes.

4.9.2 Feature-specific Guidance

None defined.

4.9.3 Use Cases

4.10 NVMe Support feature

4.10.1 Overview

This feature provides NVMe-specific use cases.

4.10.2 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](#).

4.10.3 Use Cases

- [Attach a Namespace](#)
- [Confirm valid LBA formats](#)
- [Deprovision a Namespace](#)
- [Detach a Namespace](#)
- [Can a new Namespace be created?](#)
- [Provision a Namespace with a specific LBA format](#)
- [Provision a Namespace](#)
- [Provision a Namespace from NVM Set](#)
- [Report Namespace Capacity](#)
- [Report Remaining Life for a Namespace](#)
- [Update access rights on an existing volume](#)

4.11 Replication Feature

4.11.1 Overview

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

4.11.2 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, and are grouped separately in the table below.

4.11.3 Use Cases

- [Create a New Replication Relationship by Assigning a Target Volume](#)
- [Make a New Replication Relationship by Creating a Target Volume](#)
- [Remove Replication Relationship](#)
- [Resume the Replication Synchronization Activity](#)
- [Reverse a Replication Relationship](#)
- [Split a Replica](#)
- [Suspend Replication Synchronization Activity](#)
- [Create a New Replication Relationship by Assigning an existing Target Consistency Group](#)
- [Make a New Replication Relationship by Creating a Target Consistency Group](#)
- [Remove Replication Relationship for a Consistency Group](#)
- [Resume the Replication Synchronization Activity for a Consistency Group](#)
- [Reverse a Replication Relationship for Consistency Groups](#)
- [Split a set of Replicas in Consistency Groups](#)
- [Suspend Replication Synchronization Activity between Consistency Groups](#)

5 Use cases

5.1 Alphabetic list of use cases

5.1.1 *Add Multiple Drives to an Existing Storage Pool*

Summary: Add multiple drives to an Storage Pool.

Purpose: Add multiple drives to an Storage Pool to provide additional capacity.

Triggers: Expand resources available to a pool; this could be performance, capacity or application triggered.

Detailed Context: The storage admin needs to increase the underlying available capacity within an existing pool. This use case makes the following assumptions about the “implementation” servicing the request:

- The implementation manages, or does not have any, constraints on the inclusion of multiple drives into the target storage pool.

Preconditions: User has already selected a Pool.

Feature(s): [Block provisioning](#)

Inputs:

- URL for Storage Pool: /redfish/v1/Storage/1/StoragePools/PrimaryPool
- Drives to add:

```
[{"@odata.id": "/redfish/v1/Chassis/1/Drives/1"},  
  ↪ {"@odata.id": "/redfish/v1/Chassis/1/Drives/2"}]
```

Basic Course of Events:

1. Use the “AddDrives” Action on the PrimaryPool storage pool, passing the selected drives as input.

Request: POST /redfish/v1/Storage/1/StoragePools/PrimaryPool.AddDrives

- **Headers:** Content-type : application/json
- **Body:**

```
{
  "Drives" : [
    {"@odata.id": "/redfish/v1/Chassis/1/Drives/1"},
    {"@odata.id": "/redfish/v1/Chassis/1/Drives/2"}
  ]
}
```

Response: Response is dependent on implementation's capability.

If the implementation is able to return immediately:

- **HTTP Status:** 200 (OK)
- **Headers:** None.
- **Body:** None.

If the implementation requires a background task (using the Redfish task service) to return status:

- **HTTP Status:** 202 (Accepted)
- **Headers:** Location : /redfish/v1/TaskService/Tasks/TaskID2
- **Body:** None.

Postconditions: The new drives have been added to a capacity source (chosen by the implementation) used by the selected storage pool.

Failure Scenario: None defined

See also: None defined.

5.1.2 *Attach a Namespace*

Summary: Attach a Namespace

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

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/Systems/Sys-1/Storage/SimplestNVMeSSD/
      Controllers/NVMeIOController/
```

- **Headers:** Content-type : application/json

- **Body:**

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

*

*Response:**

- **HTTP Status:** 200 (Success)
- **Headers:** Content-type : application/json
- **Body**

```
{
  "@Redfish.Copyright": "Copyright 2015-2021 SNIA. All
    ↪  rights reserved.",
  "@odata.id":
    ↪  "/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Controllers/NVMeIO
  "@odata.type":
    ↪  "#StorageController.v1_1_0.StorageController",
  "Id": "NVMeIOController",
  "Name": "NVMe I/O Controller",
  "Description": "An NVM IO controller is a general-purpose
    ↪  controller that provides access to logical block
    ↪  data and metadata stored on an NVM subsystem's
    ↪  non-volatile storage medium. IO Controllers may also
    ↪  support management capabilities.",
  "Status": {
    "State": "Enabled",
    "Health": "OK"
  },
  "Manufacturer": "Best NVMe Vendor",
  "Model": "Simple NVMe Device",
  "SerialNumber": "NVME123456",
  "PartNumber": "NVM44",
  "FirmwareVersion": "1.0.0",
  "SupportedControllerProtocols": [
    "PCIe"
  ],
  "SupportedDeviceProtocols": [
    "NVMe"
  ]
}
```

```
],
  "NVMeControllerProperties": {
    "NVMeVersion": "1.3",
    "NVMeControllerAttributes": {
      "ReportsUUIDList": false,
      "SupportsSQAssociations": false,
      "ReportsNamespaceGranularity": false,
      "SupportsTrafficBasedKeepAlive": false,
      "SupportsPredictableLatencyMode": false,
      "SupportsEnduranceGroups": false,
      "SupportsReadRecoveryLevels": false,
      "SupportsNVMSets": false,
      "SupportsExceedingPowerOfNonOperationalState":
        ↪ false,
      "Supports128BitHostId": false
    }
  },
  "Links": {
    "AttachedVolumes": [
      {
        "@odata.id":
          ↪ "/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Volume
      ]
    ]
  }
}
```

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.

5.1.3 *Can a new Namespace be created?*

Summary: Confirm that a new Namespace can be created.

Purpose: Check VolumeCollection settings before attempting to create a new entry.

Triggers: Need to create a new Namespace.

Detailed Context: Before attempting to create a new Namespace, the users wants to confirm that creation is allowed against a given StoragePool. The Allow header returned for the VolumeCollection will indicate whether Namespace creation is permitted or not.

Preconditions: None.

Feature(s): [Block provisioning](#)

Inputs:

- URL for Volumes collection: `/redfish/v1/Storage/1/Volumes`

Basic Course of Events:

1. Request the header information of the Volumes resource collection. > Note: This example use the HTTP verb HEAD, though a request using GET would also return the required headers.

Request:

HEAD `/redfish/v1/Storage/1/Volumes`

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

Response: Response contains the header.

- **HTTP Status:** 200 (OK)
- **Headers:** Allow : GET, POST, HEAD
- **Body:** None.

Postconditions: None defined.

Failure Scenario: None defined

See also: [Can a new Volume be created?](#)

5.1.4 *Can a new Namespace be created?*

Summary: Confirm that a new Namespace can be created.

Purpose: Check VolumeCollection settings before attempting to create a new entry.

Triggers: Need to create a new Namespace.

Detailed Context: Before attempting to create a new Namespace, the users wants to confirm that creation is allowed against a given StoragePool. The Allow header returned for the VolumeCollection will indicate whether Namespace creation is permitted or not.

Preconditions: None.

Feature(s): [NVMe](#)

Inputs:

- URL for Volumes collection: `/redfish/v1/Storage/1/Volumes`

Basic Course of Events:

1. Request the header information of the Volumes resource collection. > Note: This example use the HTTP verb HEAD, though a request using GET would also return the required headers.

Request:

HEAD `/redfish/v1/Storage/1/Volumes`

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

Response: Response contains the header.

- **HTTP Status:** 200 (OK)
- **Headers:** Allow : GET, POST, HEAD
- **Body:** None.

Postconditions: None defined.

Failure Scenario: None defined

See also: [Can a new Volume be created?](#)

5.1.5 *Can a new Volume be created?*

Summary: Confirm that a new Volume can be created.

Purpose: Check VolumeCollection settings before attempting to create a new Volume.

Triggers: Need to create a new Volumes.

Detailed Context: Before attempting to create a new Volume, the users wants to confirm that creation is allowed against a given StoragePool. The Allow header returned for the VolumeCollection will indicate whether Volume creation is permitted or not.

Preconditions: None.

Feature(s): [Block provisioning](#)

Inputs:

- URL for Volumes collection: /redfish/v1/Storage/1/Volumes

Basic Course of Events:

1. Request the header information of the Volumes resource collection. > Note: This example use the HTTP verb HEAD, though a request using GET would also return the required headers.

Request:

HEAD /redfish/v1/Storage/1/Volumes

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

Response: Response contains the header.

- **HTTP Status:** 200 (OK)
- **Headers:** Allow : GET, POST, HEAD
- **Body:** None.

Postconditions: None defined.

Failure Scenario: None defined

See also: [Can a new Namespace be created?](#)

5.1.6 *Change only the RAID Type of an Existing Volume*

Summary: Change only the RAID type of an existing Volume

Purpose: Change the RAID type of an existing Volume to meet new protection requirements, without a change to the underlying Drives collection.

Triggers: Change the RAID protection level of a Volume; this could be performance, capacity or application triggered.

Detailed Context: The storage admin needs to alter the underlying RAID layout for an existing Volume.

Preconditions: Volume already exists, and employs drives sufficient to satisfy the requested RAID geometry.

Feature(s): [Block provisioning](#)

Inputs:

- URL for Volume: /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolume

Basic Course of Events:

1. Use the ChangeRAIDLAYOUT Action on the Volume, passing the requested RAID-Type as input.

Request:

POST /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1/Action/ChangeRAIDLAYOUT

- **Headers:** Content-type: application/json

- **Body:**

```
{  
  "RAIDType" : "RAID5"  
}
```

Response: Response is dependent on implementation's capability.

Case 1: If the implementation is able to return immediately:

- **HTTP Status:** 204 (No Content)

- **Headers:**

Location :

/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1

- **Body:** None.

A GET on volume will show the updated RAIDType.

Request:

GET /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1

Response:

- **HTTP Status:** 200 (OK)

- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1
Content-type: application/json ETag: "FD87EC46239"

- **Body:**

```
{
  "@SSM.Copyright": "Copyright (c) 2014-2024 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_6_0.Volume",
  "Name": "MyVolume",
  "Id": "1",
  "Description": "Default Volume Description",
  "RAIDType": "RAID5",
  "Identifiers": [
    {
      "DurableNameFormat": "NAA",
      "DurableName":
        ↪ "65456765456761001234076100123487"
    }
  ],
}
```

```
"Status": {
  "State": "Enabled",
  "Health": "OK"
},
"CapacityBytes": 1099511627776,
"Links": {
  "StoragePool":
    ↪ "/redfish/v1/Storage/1/StoragePools/1"
}
}
```

Case 2: If the implementation requires a background task (using the Redfish task service) to return status:

- **HTTP Status:** 202 (Accepted)
- **Headers:** Location : /redfish/v1/TaskService/Tasks/TaskID2
ETag: "FD87EC46239"
- **Body:** None.

A GET on volume while task is pending will indicate the in-process change to RAIDType, and that the transition to the new RAID layout is not complete, using Status.State and Status.Health:

Request: GET /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVol

Response:

- **HTTP Status:** 200 (OK)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/Al
Content-type: application/json ETag: "FD87EC46239"
- **Body:**

```
{
  "@SSM.Copyright": "Copyright (c) 2014-2024
    ↪ SNIA. All rights reserved.",
  "@odata.context":
    ↪ "/redfish/v1/$metadata#Volume.Volume",
}
```

```

"@odata.id":
  ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVol
"@odata.type": "#Volume_1_6_0.Volume",
"Name": "MyVolume",
"Id": "1",
"Description": "Default Volume Description",
"RAIDType": "RAID5",
"Identifiers": [
{
  "DurableNameFormat": "NAA",
  "DurableName":
    ↪ "65456765456761001234076100123487"
}
],
"Status": {
  "State": "Updating",
  "Health": "Warning"
},
"CapacityBytes": 1099511627776,
"Links": {
  "StoragePool":
    ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool"
}
}

```

A subsequent GET on the Volume, once the task has completed, will reflect the new values:

Request:

GET /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1

Response:

- **HTTP Status:** 200 (OK)
- **Headers:**

Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVol
 Content-type: application/json ETag: FD87EC46781

- Body:

```
{
  "@SSM.Copyright": "Copyright (c) 2014-2024 SNIA.
    ↪ All rights reserved.",
  "@odata.context":
    ↪ "/redfish/v1/$metadata#Volume.Volume",
  "@odata.id":
    ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolume",
  "@odata.type": "#Volume_1_6_0.Volume",
  "Name": "MyVolume",
  "Id": "1",
  "Description": "Default Volume Description",
  "RAIDType": "RAID5",
  "Identifiers": [
    {
      "DurableNameFormat": "NAA",
      "DurableName":
        ↪ "65456765456761001234076100123487"
    }
  ],
  "Status": {
    "State": "Enabled",
    "Health": "OK"
  },
  "CapacityBytes": 1099511627776,
  "Links": {
    "StoragePool":
      ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool"
  }
}
```

Postconditions: None defined.

Failure Scenario: If the system is unable to complete the requested change to the RAID layout for some reason (e.g., insufficient Drives in the underlying StoragePool to support the requested RAID type), the initial POST will result in an error. For example:

1. Use the ChangeRAIDLayout Action on the Volume, passing the requested RAID-Type as input, as in the initial scenario.

Request:

POST /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1/Actions/ChangeRAIDLayout

- **Headers:** Content-type : application/json

- **Body:**

```
{  
  "RAIDType" : "RAID5"  
}
```

Response:

- **HTTP Status:** 400 (Bad Request)
- **Headers:**

Content-type: application/json

- **Body:**

```
{  
  "error": {  
    "code": "Base.1.6.ActionParameterMissing",  
    "message": "The action ChangeRAIDLayout requires the  
      ↪ parameter Drives to be present in the request  
      ↪ body.",  
    "@Message.ExtendedInfo" : [  
      {  
        "MessageId" : "Base.1.6.ActionParameterMissing",  
        "Message" : "The Drives paramter must be included  
          ↪ in this request",  
        "RelatedProperties" : "Drives"  
      }  
    ]  
  }  
}
```

See also: None defined.

5.1.7 *Change only the span count of an existing volume*

Summary: Change only the span count of an existing Volume

Purpose: Change the span count RAID type of an existing Volume, without a change to the underlying Drives collection.

Triggers: Change the span count of a Volume; this could be performance, capacity or application triggered.

Detailed Context: The storage admin needs to alter the underlying RAID layout for an existing Volume.

Preconditions: Volume already exists, and employs drives sufficient to satisfy the requested RAID geometry.

Feature(s): [Block provisioning](#)

Inputs:

- URL for Volume: /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolume

Basic Course of Events:

1. Use the ChangeRAIDLayout Action on the Volume, passing the requested MediaSpanCount as input.

Request:

POST /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1/
Actions/ChangeRAIDLayout

- **Headers:** Content-type : application/json

- **Body:**

```
{  
  "MediaSpanCount" : 10  
}
```

Response:

Response is dependent on implementation's capability.

If the implementation is able to return immediately:

- **HTTP Status:** 204 (No Content)

- **Headers:**

- Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1

- **Body:** None.

A GET on volume will show the updated RAIDType.

Request:

`GET /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1`

- **Response:** 200 (OK)

- **Headers:**

- Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1

- Content-type : application/jsonETag: FD87EC46781

- **Body:**

```
{
  "@SSM.Copyright": "Copyright (c) 2014-2024 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_6_0.Volume",
  "Name": "MyVolume",
  "Id": "1",
  "Description": "Default Volume Description",
  "RAIDType": "RAID50",
  "MediaSpanCount": 10,
  "Identifiers": [
    {
      "DurableNameFormat": "NAA",
      "DurableName": "65456765456761001234076100123487"
```

```

    }
  ],
  "Status": {
    "State": "Enabled",
    "Health": "OK"
  },
  "CapacityBytes": 1099511627776,
  "Links": {
    "StoragePool":
      ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool"
  }
}

```

If the implementation requires a background task (using the Redfish task service) to return status:

Response:

- **HTTP Status:** 202 (Accepted)
- **Headers:**
 - Location : /redfish/v1/TaskService/Tasks/TaskID2
- **Body:** None.

A GET on volume while task is pending will indicate the in-process change to RAIDType, and that the transition to the new RAID layout is not complete, using Status.State and Status.Health:

Request:

```
`GET /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1`
```

Response:

- **HTTP Status:** 200 (OK)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1
Content-type : application/json ETag: FD87EC46781
- **Body:**

```
{
  "@SSM.Copyright": "Copyright (c) 2014-2024 SNIA. All
    ↪  rights reserved.",
  "@odata.context":
    ↪  "/redfish/v1/$metadata#Volume.Volume",
  "@odata.id":
    ↪  "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/
  "@odata.type": "#Volume_1_6_0.Volume",
  "Name": "MyVolume",
  "Id": "1",
  "Description": "Default Volume Description",
  "RAIDType": "RAID50",
  "MediaSpanCount": 10,
  "Identifiers": [
    {
      "DurableNameFormat": "NAA",
      "DurableName": "65456765456761001234076100123487"
    }
  ],
  "Status": {
    "State": "Updating",
    "Health": "Warning"
  },
  "CapacityBytes": 1099511627776,
  "Links": {
    "StoragePool":
      ↪  "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/
  }
}
```

A subsequent GET on the Volume, once the task has completed, will reflect the new values:

Request:

```
`GET /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1`
```

Response:

- **HTTP Status:** 200 (OK)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/Alloc
Content-type : application/json ETag: FD87EC469AE
- **Body:**

```
{
  "@SSM.Copyright": "Copyright (c) 2014-2024 SNIA. All
    ↪ rights reserved.",
  "@odata.context":
    ↪ "/redfish/v1/$metadata#Volume.Volume",
  "@odata.id":
    ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/",
  "@odata.type": "#Volume_1_6_0.Volume",
  "Name": "MyVolume",
  "Id": "1",
  "Description": "Default Volume Description",
  "RAIDType": "RAID50",
  "MediaSpanCount": 10,
  "Identifiers": [
    {
      "DurableNameFormat": "NAA",
      "DurableName": "65456765456761001234076100123487"
    }
  ],
  "Status": {
    "State": "Enabled",
    "Health": "OK"
  },
  "CapacityBytes": 1099511627776,
  "Links": {
    "StoragePool":
      ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool"
  }
}
```

Postconditions: None defined.

Failure Scenario: If the system is unable to complete the requested change to the RAID layout for some reason (e.g., insufficient Drives in the underlying StoragePool to support the requested MediaSpanCount), the initial POST will result in an error. For example:

1. Use the ChangeRAIDLayout Action on the Volume, passing the requested MediaSpanCount as input, as in the initial scenario.

Request:

POST /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1/
Actions/ChangeRAIDLayout

- **Headers:** Content-type : application/json

- **Body:**

```
{  
  "MediaSpanCount": 10  
}
```

Response:

- **HTTP Status:** 400 (Bad Request)

- **Headers:** Content-type : application/json

- **Body:**

```
{  
  "error": {  
    "code": "Base.1.6.ActionParameterMissing",  
    "message": "The action ChangeRAIDLayout requires the  
    ↪ parameter Drives to be present in the request  
    ↪ body.",  
    "@Message.ExtendedInfo" : [  
      "MessageId": "Base.1.6.ActionParameterMissing",  
      "Message" : "The Drives paramter must be included  
    ↪ in this request",  
      "RelatedProperties" : "Drives"  
    ]  
  }  
}
```

```
}  
}
```

See also: None defined.

5.1.8 Confirm valid LBA formats

Summary: Confirm valid LBA formats

Purpose: Verify that a given LBA format is supported by a Volume collection.

Triggers: None defined.

Detailed Context: Querying the CollectionCapabilities object defined within the Controllers collection, allows a user to confirm that a particular LBA format is supported, even if the collection itself is empty due to the ephemeral nature of NVMe Controllers.

Preconditions: None defined.

Feature(s): [NVMe](#)

Inputs:

- URL for Controller collection: `/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Controllers`

Basic Course of Events:

1. Get the ControllerCollection and find the CapabilitiesObject property within the returned JSON data:

Request:

GET `/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Controllers`

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

Response:

- **HTTP Status:** 200 (OK)
- **Headers:**
 - Content-type : application/json
 - ETag : "97AED48652"
- **Body:**

```
{
  "@odata.type":
    "#StorageControllerCollection.StorageControllerCollection",
```



```
"Name": "Storage Controller Collection",
"Description": "Storage Controller Collection",
"Members@odata.count": 0,
"Members": [],
"@Redfish.CollectionCapabilities": {
  "@odata.type":
    ↪ "#CollectionCapabilities.v1_3_0.CollectionCapabilities",
  "Capabilities": [{
    "CapabilitiesObject": {
      "@odata.id":
        ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/Capabilities",
    },
    "Links": {
      "TargetCollection": {
        "@odata.id":
          ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Controllers"
      }
    }
  }]
},
"@odata.id":
  ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Controllers",
"@Redfish.Copyright": "Copyright 2022 SNIA. All rights
  ↪ reserved."
}
```

2. Get the CollectionCapabilities, and confirm that the desired LBA format is included in the collection of supported formats.

Request:

GET /redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/Capabilities

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

Response:

- **HTTP Status:** 200 (OK)
- **Headers:**
 - Content-type : application/json
 - ETag : "1AB7994D"
- **Body:**

```
{
"@odata.type": "#Volume.v1_9_0.Volume",
"Id": "Capabilities",
"Name": "Capabilities for the Volume",
"Name@Redfish.RequiredOnCreate": true,
"Name@Redfish.SetOnlyOnCreate": true,
"Description@Redfish.OptionalOnCreate": true,
"Description@Redfish.SetOnlyOnCreate": true,

"NVMeNamespaceProperties": {
  "LBAFormatsSupported@Redfish.AllowableValues": [
    "LBAFormat0",
    "LBAFormat1",
  ],
  "LBAFormats": [
    {
      "LBAFormatType": "LBAFormat0",
      "RelativePerformance": "Best",
      "LBADataSizeBytes": 4096,
      "LBAMetadataSizeBytes": 0
    },
    {
      "LBAFormatType": "LBAFormat1",
      "RelativePerformance": "Good",
      "LBADataSizeBytes": 512,
      "LBAMetadataSizeBytes": 0
    }
  ]
},
},
```

```
"@odata.id":  
  ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/Capabilities",  
"@Redfish.Copyright": "Copyright 2022 SNIA. All rights  
  ↪ reserved."  
}
```

Postconditions: None.

Failure Scenario: None defined.

See also: None defined.

5.1.9 Create a new connection to an existing volume

Summary: Create a new connection to an existing Volume, using a pre-existing Endpoint to grant read/write access to the Volume.

Purpose: Use existing Endpoints to provide a new access path to an existing volume.

Triggers: None defined.

Detailed Context: Create a new connection to an existing Volume, to allow it to be accessed from a given host by way of the selected Endpoints.

Preconditions: User has already identified the Endpoints that will be used for the new Connection.

Feature(s): [Access Management](#)

Inputs:

- Name for new connection: "Connection info for host 1"
- Volume Id: "/redfish/v1/Storage/IPAttachedDrive1/Volumes/SimpleNamespace"
- Initiator endpoint: /redfish/v1/Fabrics/NVMeoF/Endpoints/Initiator1
- Target endpoint: /redfish/v1/Fabrics/NVMeoF/Endpoints/D1-E1
- Volume access information:

```
{
  "AccessCapabilities": [
    "Read",
    "Write"
  ],
  "Volume": {
    "@odata.id":
      ↪ "/redfish/v1/Storage/IPAttachedDrive1/Volumes/SimpleNames
  }
}
```

Basic Course of Events:

1. Create the new Connection

Request:

POST /redfish/v1/Fabrics/NVMeoF/Connections

- **Headers:** Content-type : application/json
- **Body:**

```
{
  "Name": "Connection info for host 1",
  "VolumeInfo": [
    {
      "AccessCapabilities": [
        "Read",
        "Write"
      ],
      "Volume": {
        "@odata.id":
          ↪ "/redfish/v1/Storage/IPAttachedDrive1/Volumes/SimpleNames
      }
    },
  ],
  "Links": {
    "InitiatorEndpoints": [
      {
        "@odata.id":
          ↪ "/redfish/v1/Fabrics/NVMeoF/Endpoints/Initiator1"
      }
    ],
    "TargetEndpoints": [
      {
        "@odata.id":
          ↪ "/redfish/v1/Fabrics/NVMeoF/Endpoints/D1-E1"
      }
    ]
  }
}
```

Response:

- **HTTP Status:** 201 (Created)

- **Headers:**

Content-type: application/json

Location: /redfish/v1/Fabrics/NVMeoF/Connections/1

- **Body:**

```
{
  "Name": "Connection info for host 1",
  "VolumeInfo": [
    {
      "AccessCapabilities": [
        "Read",
        "Write"
      ],
      "Volume": {
        "@odata.id":
          ↪ "/redfish/v1/Storage/IPAttachedDrive1/Volumes/SimpleN
      },
    ],
    "Links": {
      "InitiatorEndpoints": [
        {
          "@odata.id":
            ↪ "/redfish/v1/Fabrics/NVMeoF/Endpoints/Initiator1"
        },
      ],
      "TargetEndpoints": [
        {
          "@odata.id":
            ↪ "/redfish/v1/Fabrics/NVMeoF/Endpoints/D1-E1"
        },
      ],
    }
  ]
}
```

Postconditions: None defined.

Failure Scenario: None defined

See also: None defined.

5.1.10 *Create a new endpoint*

Summary: Create a new Endpoint.

Purpose: Create an Endpoint to direct access to a system's resources to a particular interface and protocol.

Triggers: None defined.

Detailed Context: Create an Endpoint to direct access to a system's resources to a particular interface and protocol.

Preconditions: None.

Feature(s): [Access Management](#)

Inputs:

- Endpoint role: "Initiator"
- Supported protocol: "iSCSI",
- Related transport details:

```
json { "IPv4Address": {  
  "Address": "192.168.1.63" } }
```

Basic Course of Events:

1. Create the new Endpoint

Request:

POST /redfish/v1/Storage/MidrangeStorageSystem/Endpoints

- **Headers:** Content-type : application/json

- **Body:**

```
{  
  "Name": "InitiatorEndpoint1"  
  "EntityRole": "Initiator"  
  "EndpointProtocol": "iSCSI"  
  "IPTransportDetails":  
  {  
    "IPv4Address": {  
      "Address": "192.168.1.63"  
    }  
  }  
}
```



```
}  
}
```

Response:

- **HTTP Status:** 201 (Created)

- **Headers:**

Content-type: application/json

Location: /redfish/v1/Storage/MidrangeStorageSystem/EndpointGroups/Tar

- **Body:**

```
{  
  "@odata.type": "#Endpoint.v1_7_0.Endpoint",  
  "Id": "InitiatorEndpoint1",  
  "Name": "InitiatorEndpoint1",  
  "Description": "iSCSI target Endpoint 1",  
  "Identifiers": [  
    {  
      "DurableName":  
        ↪ "4289a3dd-ba52-41b8-9d3d-38de196e059c",  
      "DurableNameFormat": "UUID"  
    }  
  ],  
  "ConnectedEntities": [  
  ],  
  "EndpointProtocol": "iSCSI",  
  "IPTransportDetails": [  
    {  
      "IPv4Address": {  
        "Address": "192.168.1.63"  
      }  
    }  
  ],  
  "Status": {  
    "State": "Enabled",
```

```
        "Health": "OK"
    },
    "@odata.id":
        ↪ "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/InitiatorEnd
    "@Redfish.Copyright": "Copyright 2015-2024 SNIA. All
        ↪ rights reserved."
}
```

Postconditions: None defined.

Failure Scenario: None defined

See also: None defined.

5.1.11 Create a new endpoint group

Summary: Create a new EndpointGroup, using a pre-existing Endpoints.

Purpose: Create an EndpointGroup to simplify future management of a group of existing Endpoints.

Triggers: None defined.

Detailed Context: Grouping a set of existing Endpoints into a single EndpointGroup simplifies access management by allowing multiple Endpoints to be managed with a single API request.

Preconditions: User has already identified the Endpoints that will included in the new EndpointGroup.

Feature(s): [Access Management](#)

Inputs:

- A set of pre-existing Endpoints:

```
...
[
  {
    "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/TargetEndpoint1",
  },
  {
    "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/TargetEndpoint2",
  },
  {
    "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/TargetEndpoint3",
  },
  {
    "@odata.id": "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/TargetEndpoint4",
  }
]
...
```

- Name of new EndpointGroup: "TargetEndpoints"

- Type of Endpoints in new EndpointGroup: “Target”

Basic Course of Events:

1. Create the new EndpointGroup

Request:

POST /redfish/v1/Storage/MidrangeStorageSystem/EndpointGroups

- **Headers:** Content-type : application/json

- **Body:**

```
{
  "Name": "TargetEndpoints"
  "GroupType": "Target"
  "Endpoints": [
    {
      "@odata.id":
        ↪ "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/TargetEn
    },
    {
      "@odata.id":
        ↪ "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/TargetEn
    },
    {
      "@odata.id":
        ↪ "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/TargetEn
    },
    {
      "@odata.id":
        ↪ "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/TargetEn
    }
  ]
}
```

Response:

- **HTTP Status:** 201 (Created)
- **Headers:**

Content-type: application/json

Location: /redfish/v1/Storage/MidrangeStorageSystem/EndpointGroups/Tar

• **Body:**

```
{
  "@odata.type":
    ↪ "#EndpointGroup.v1_2_0.EndpointGroup",
  "Name": "TargetEndpoints"
  "Id": "TargetEndpoints"
  "Description": "Group of target endpoints for the
    ↪ midrange storage system",
  "AccessState": "Optimized",
  "TargetEndpointGroupIdentifier": 3,
  "GroupType": "Server",
  "Endpoints": [
    {
      "@odata.id":
        ↪ "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/Targ
    },
    {
      "@odata.id":
        ↪ "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/Targ
    },
    {
      "@odata.id":
        ↪ "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/Targ
    },
    {
      "@odata.id":
        ↪ "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/Targ
    }
  ],
  "@odata.id":
    ↪ "/redfish/v1/Storage/MidrangeStorageSystem/EndpointGroups/Tar
```

```
    "@Redfish.Copyright": "Copyright 2015-2024 SNIA.  
    ↪ All rights reserved."  
  }
```

Postconditions: None defined.

Failure Scenario: None defined

See also: None defined.

5.1.12 *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.

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:** Content-type : application/json

- **Body:**

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

Response:

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/ConsistencyGroup/CG_DB1
- **Body:** None.

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](#).

5.1.13 Create a New Replication Relationship by Assigning a Target Volume

Summary: Establish a replication relationship by assigning an existing volume to serve as a target replica for an existing source volume.

Purpose: Leverage an existing volume resource to provide expanded data protection through a replica relationship with the specified source volume.

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

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

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

Feature(s): [Replication \(both local and remote\)](#)

Inputs:

- URLfortargetvolume: /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedV
- Requested replica type: Mirror
- 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.AssignReplicaTarget

- **Headers:** Content-type : application/json

- **Body:**

```
{  
  "ReplicaUpdateMode": "Synchronous",  
  "TargetVolume":  
    ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/650  
  "ReplicaType": "Mirror"  
}
```

Response:

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/Alloc
- **Body:** None.

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.

Failure Scenario: None defined

See also: [Assign Replica Target \(CG\)](#).

5.1.14 *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.

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/Allocat

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:**
None

Response:

- **HTTP Status:** 200 (Success)

- **Headers:**

Content-type : application/json ETag : "97AED48652"

- **Body:**

```
{
  "SupportedLinesOfService": [
    {
      "@odata.id":
        ↪ "/redfish/v1/StorageServices/1/Links/DataProtectionLoSCapabilities/
      "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:**

Content-type : application/json ETag : "97AED48652"

- **Body:**

```
{
  "ReplicaRequests": [{
    "ReplicaSource": {"@odata.id":
      ↪ "/redfish/v1/StorageServices/1/StoragePools/BasePool/Allocate
    "ReplicaName": "MySnapshot"
  }]
}
```

Response:

- **HTTP Status:** 200 (Success)

- **Headers:**

- Content-type : application/json
- ETag : "97AED48881"

- **Body:**

```
[ {  
  "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/AllocatedV
```

```
    "DataProtectionLineOfService":  
      ↪ "/redfish/v1/StorageServices/1/Links/DataProtectionLoSCapabilit  
  ]
```

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.

5.1.15 *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

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 from 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
 - * "AverageIOOperationLatencyMicroseconds": 10

Basic Course of Events:

1. Create ClassOfService

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

- **Headers:** Content-type : application/json
- **Body:**

```
{  
  "Name": "SSD",
```

```
"Description": "Minimal SSD class of service.",
"LinesOfService": {
  "IOPerformanceLinesOfService": [ {
    "Name": "SSDLoS",
    "I0OperationsPerSecondIsLimited": false,
    "MaxI0OperationsPerSecondPerTerabyte": 1000000,
    "AverageI0OperationLatencyMicroseconds": 10
  }]
}
```

Response:

- **HTTP Status:** 201 (Created)
- **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.

5.1.16 Create ConsistencyGroup

Summary: Create a ConsistencyGroup

Purpose: Create a ConsistencyGroup

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/Storage
- The proposed name of the new ConsistencyGroup (string): "HillOfBeans1"
- A description of the new ConsistencyGroup (string): "New storage group to support accounting"
- A consistency requirement for the new ConsistencyGroup (boolean). When TRUE, all copies of data within a ConsistencyGroup must be kept in sync: true
- Pointers to the volumes that are members of this group.

Basic Course of Events:

1. Create ConsistencyGroup

Request:

POST /redfish/v1/StorageService/StorageGroups/Members

- **Headers:** Content-type : application/json

- **Body:**

```
{
  "Name" : "HillOfBeans1",
  "Description" : "New consistency group to support
    ↪ accounting",
  "IsConsistent" : true,
  "ConsistencyType": "ApplicationConsistent",
  "ConsistencyMethod": "Other",
```

```
"Volumes": [  
  {  
    "@odata.id":  
      ↪ "/redfish/v1/Storage/MySystem/StoragePools/ConsistencyPool/Alloca  
  },  
  {  
    "@odata.id":  
      ↪ "/redfish/v1/Storage/MySystem/StoragePools/ConsistencyPool/Alloca  
  },  
  {  
    "@odata.id":  
      ↪ "/redfish/v1/Storage/MySystem/StoragePools/ConsistencyPool/Alloca  
  }  
]  
}
```

Response:

```
{  
  "Name" : "HillOfBeans1",  
  "Description" : "New consistency group to support  
    ↪ accounting",  
  "IsConsistent" : true,  
  "ConsistencyType": "ApplicationConsistent",  
  "ConsistencyMethod": "Other",  
  "Status": {  
    "State": "Enabled"  
  },  
  "Volumes": [  
    {  
      "@odata.id":  
        ↪ "/redfish/v1/Storage/MySystem/StoragePools/ConsistencyPool/Allocate  
    },  
    {  
      "@odata.id":  
        ↪ "/redfish/v1/Storage/MySystem/StoragePools/ConsistencyPool/Allocate
```

```
    },  
    {  
      "@odata.id":  
        ↪ "/redfish/v1/Storage/MySystem/StoragePools/ConsistencyPool/Allocate  
    }  
  ]  
}
```

- **HTTP Status:** 200 (OK)

Postconditions: The requested ConsistencyGroup is added to the Storage instance.

Failure Scenario: None defined.

See also: None defined.

5.1.17 Create file share

Summary: Create a file share

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

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:** Content-type : application/json
- **Body:**

```
{  
  "Name" : "MyShare",  
  "Description" : "Share of files under MyShare.",  
  "SharedFilePath" : "/Shares/MyShare"  
}
```

Response:

- **HTTP Status:** 201 (Created)
- **Headers:**

Location : /redfish/v1/StorageServices/1/FileSystems/QuickFiles/ExportedShares/MyShare
Content-type : application/json

- **Body:**

```
{
  "@Redfish.Copyright": "Copyright 2015-2021 SNIA. All rights
    ↪ reserved.",
  "@odata.id":
    ↪ "/redfish/v1/StorageServices/1/FileSystems/QuickFiles/ExportedShares/MyS",
  "@odata.type": "#FileShare.v1_0_0.FileShare",
  "Id": "MyShare",
  "Name": "MyShare",
  "Description": "My File Share.",
  "FileSharePath": "/Shares/MyShare",
  "FileSharingProtocols": [
    "SMBv3_1_1"
  ],
  "Status": {
    "State": "Enabled",
    "Health": "OK",
    "HealthRollup": "OK"
  },
  "DefaultAccessCapabilities": [
    "Read",
    "Write",
    "Execute"
  ],
  "ExecuteSupport": true,
  "RootAccess": true,
  "CASupported": true,
  "EthernetInterfaces": {
    "@odata.id":
      ↪ "/redfish/v1/Systems/FileServer/EthernetInterfaces/1"
  },
  "Links": {
    "FileSystem": {
      "@odata.id":
        ↪ "/redfish/v1/StorageServices/1/FileSystems/QuickFiles"
```

```
}  
}  
}
```

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

Failure Scenario: None defined.

See also: None defined.

5.1.18 Create file system

Summary: Create a file system

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

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": 1000000000000000,
    "IsThinProvisioned": true
  }
}
```

Basic Course of Events:

1. Create FileSystem

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

- **Headers:** Content-type : application/json
- **Body:**

```
{
  "Name": "QuickFiles",
```

```
"Description": "100 TB FileSystem having SSD class
  ↪ storage.",
"Capacity": {
  "Data": {
    "ProvisionedBytes": 1000000000000000
  },
  "IsThinProvisioned": true
},
"Links" : {
  "ClassOfService": {"odata.id":
    ↪ "/redfish/v1/StorageServices/1/Links/ClassesOfService/SSD"}
}
```

Response:

- **HTTP Status:** 201 (Created)

- **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.

5.1.19 Create file system

Summary: Create a file system

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

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": 1000000000000000;
    "IsThinProvisioned": true;
  }
}
```

Basic Course of Events:

1. Create FileSystem

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

- **Headers:** Content-type : application/json
- **Body:**

```
{
  "Name": "QuickFiles",
```

```
"Description": "100 TB FileSystem having SSD class
  ↳ storage.",
"Capacity": {
  "Data": {
    "ProvisionedBytes": 1000000000000000
  },
  "IsThinProvisioned": true
},
"Links" : {
  "ClassOfService": {"odata.id":

  ↳ "/redfish/v1/StorageServices/1/Links/ClassesOfService/S

}
}
```

Response:

- **HTTP Status:** 201 (Created)

- **Headers:**

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

Content-type : application/json

- **Body:**

```
{
"@Redfish.Copyright": "Copyright 2015-2024 SNIA. All
  ↳ rights reserved.",
"@odata.id":
  ↳ "/redfish/v1/StorageServices/FileService/FileSystems/QuickFiles",
"@odata.type": "#FileSystem.v1_1_0.FileSystem",
"Id": "QuickFiles",
"Name": "QuickFiles",
"Description": "QuickFiles FileSystem, not
  ↳ replicated.",
"BlockSizeBytes": 8192,
"Capacity": {
  "Data": {
```

```
        "ConsumedBytes": 0,
        "AllocatedBytes": 0,
        "ProvisionedBytes": 10000000000000
    },
    "Metadata": {
        "ConsumedBytes": 0,
        "AllocatedBytes": 0,
        "ProvisionedBytes": 0
    },
    "Snapshot": {
        "ConsumedBytes": 0,
        "AllocatedBytes": 0,
        "ProvisionedBytes": 0
    }
},
"CapacitySources": [
    {
        "@odata.id":
        ↪ "/redfish/v1/StorageServices/FileService/FileSystems/QuickFi
    ]
},
"LowSpaceWarningThresholdPercents": [
    60,
    90
],
"AccessCapabilities": [
    "Read",
    "Write"
],
"CaseSensitive": false,
"CasePreserved": false,
"CharacterCodeSet": [
    "ASCII",
    "Unicode"
],
```

```
"MaxFileNameLengthBytes": 256,  
"ClusterSizeBytes": 256,  
"Links": {  
  "ClassOfService": {  
    "@odata.id":  
      ↪ "/redfish/v1/StorageServices/FileService/ClassesOfService/Go  
  }  
}
```

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

Failure Scenario: None defined.

See also: None defined.

5.1.20 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.

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.

Response:

- **HTTP Status:** 200 (OK)

- **Headers:**

ETag: "123-a" Content-type : application/json

- **Body:**

```
{
  "Value": [{
    "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" Content-type : application/json

- **Body:**

```
{
  "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:

- **HTTP Status:** 204 (No Content)
- **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.

5.1.21 Create storage pool and specify a pool type

Summary: Create a StoragePool and select SupportedPoolTypes

Purpose: In addition to creating a StoragePool, this query sets a value for the optional property SupportedPoolTypes.

Triggers: Users need to allocate storage.

Detailed Context: Create a storage pool containing an amount of storage that can be used to create volumes. In this case, the query includes a value for SupportedPoolTypes, which can only be set by a client during pool creation.

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

Feature(s): None.

Inputs:

- URL for storage pool collection:
 - /redfish/v1/Storage/1/StoragePools
 - Alternate: /redfish/v1/StorageServices/1/StoragePools
- Name for the new storage pool: "Storage Pool 1"
- Id for the new storage pool: "SP1"
- Description: "Storage Pool for block storage"
- Set of Drives, and additional parameter to indicate supported pool type(s):

```
{
  "Drives": [
    {"odata.id": "/redfish/v1/Chassis/1/Drives/1"}
  ],
  "SupportedPoolTypes": [ "Block" ]
}
```

Basic Course of Events:

1. Create StoragePool

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

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

- **Body:**

```
{
  "Name": "Storage Pool 1",
  "Id": "SP1",
  "Description": "Storage Pool for block storage",
  "Drives": [
    {"odata.id": "/redfish/v1/Chassis/1/Drives/1"}
  ],
  "SupportedPoolTypes": [
    "Block"
  ]
}
```

Response:

- **HTTP Status:** 201 (Created)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/SP1
- **Body:**

```
{
  "@Redfish.Copyright": "Copyright 2015-2024 SNIA. All
  ↪ rights reserved.",
  "@odata.id": "/redfish/v1/Storage/1/StoragePools/SP1",
  "@odata.type": "#StoragePool.v1_6_0.StoragePool",
  "Id": "SP1",
  "Name": "SSD",
  "Description": "Storage Pool 1",
  "RemainingCapacityPercent": 100,
  "Capacity": {
    "Data": {
      "AllocatedBytes": 600135780272,
      "ConsumedBytes": 0
    }
  },
}
```

```
"CapacitySources": [  
  {  
    "@odata.id":  
      ↪ "/redfish/v1/Storage/1/StoragePools/SP1/CapacitySources/Source1",  
  },  
],  
"Links": {  
  "OwningStorageResource": {  
    "@odata.id": "/redfish/v1/Storage/1"  
  }  
}
```

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

Failure Scenario: None defined.

See also: [Create storage pool using specified set of drives](#)

5.1.22 *Create storage pool*

Summary: Create a StoragePool

Purpose: Create a StoragePool

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

```
{
  "Data": {
    "ProvisionedBytes": 1000000000000000,
    "IsThinProvisioned": false
  }
}
```

Basic Course of Events:

1. Create StoragePool

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

- **Headers:** Content-type : application/json
- **Body:**

```
{
  "Name": "SSD",
  "Description": "100 TB pool of SSD class storage.",
  "Capacity": {
    "Data": {
      "ProvisionedBytes": 1000000000000000
    },
    "IsThinProvisioned": false
  },
  "ClassesOfService": {
    "Members": [{
      "@odata.id":
        ↪ "/redfish/v1/StorageServices/1/ClassesOfService/SSD"
    }]
  }
}
```

Response:

- **HTTP Status:** 201 (Created)
- **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.

5.1.23 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

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. This use case expects that the implementation will automatically create the CapacitySource object specified as part of the StoragePool object creation process.

Preconditions: The user has the information about the set of drives. These will be used to create a new CapacitySource named "CapacitySource1".

Feature(s): None.

Inputs:

- URL for storage pool collection:
 - /redfish/v1/Storage/1/StoragePools
- Name for the new storage pool: "SSD"
- Description: "SSD Storage Pool"
- Set of Drives:

```
{
  "CapacitySource1": [{
    "ProvidingDrives": [
      {"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"],  
  "CapacitySource1": [{  
    "ProvidingDrives": [  
      {"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:

- **HTTP Status:** 201 (Created)

- **Headers:** Location : redfish/v1/Storage/1/StoragePools/SP111

- **Body:**

```
{
"@Redfish.Copyright": "Copyright 2015-2024 SNIA. All
↪  rights reserved.",
"@odata.id": "/redfish/v1/Storage/1/StoragePools/SP111",
"@odata.type": "#StoragePool.v1_6_0.StoragePool",
"Id": "SP111",
"Name": "SSD",
"Description": "Storage Pool for RAID1, 5 or 50.",
"SupportedRAIDTypes": [
    "RAID1",
    "RAID5",
    "RAID50"
],
"RemainingCapacityPercent": 100,
"Capacity": {
    "Data": {
        "AllocatedBytes": 600135780272,
        "ConsumedBytes": 0
    }
},
"CapacitySources": [
    {
        "@odata.id":
        ↪  "/redfish/v1/Storage/1/StoragePools/SP111/CapacitySources/Capac
    }
],
"Links": {
    "OwningStorageResource": {
        "@odata.id": "/redfish/v1//Storage/1"
    }
}
}
```

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

Failure Scenario: None defined.

See also: None defined.

5.1.24 Create storage pool using specified set of drives

Summary: Create a StoragePool

Purpose: Create a StoragePool using a specified set of drives

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:

```
{
  "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:**

```

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

```

Response:

- **HTTP Status:** 201 (Created)

- **Headers:**

Location : /redfish/v1/Storage/1/StoragePools/SP111

- **Body:** “json { “@Redfish.Copyright”: “Copyright 2015-2024 SNIA. All rights reserved.”, “@odata.id”: “/redfish/v1/Storage/1/StoragePools/SP111”, “@odata.type”: “#StoragePool.v1_6_0.StoragePool”, “Id”: “SP111”, “Name”: “SSD”, “Description”: “Storage Pool 1”, “RemainingCapacityPercent”: 100, “Capacity”: { “Data”: { “AllocatedBytes”: 600135780272, “ConsumedBytes”: 0 } }, “CapacitySources”: [{ “@odata.id”: “/redfish/v1/Storage/1/StoragePools/SP111/CapacitySources/Source1” }], “Links”: { “OwningStorageResource”: { “@odata.id”: “/redfish/v1//Storage/1” } } }

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

****Failure Scenario:**** None defined.

****See also:**** None defined.

\pagebreak

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.

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

****Detailed Context:**** The admin needs to satisfy a user or application request to

****Preconditions:**** User has already selected a Pool and checked the supported RAID

****Feature(s):**** [Block provisioning](#block-provisioning-feature)

****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 AllocatedVolumes resource collection.

This instructs the service to use the identified StoragePool to allocate a new volume 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/AllocatedVolumes`

- ****Headers:**** `Content-type : application/json`

- ****Body:****

```
```json
{
 "Name" : "MyVolume",
```

```
"CapacityBytes" : 1099511627776,
"RAIDType" : "RAID1"
}
```

**Response:** Response contains the details of the created volume.

- **HTTP Status:** 201 (Created)
- **Headers:**

Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1  
Content-type : application/json

- **Body:**

```
{
 "@SSM.Copyright": "Copyright (c) 2014-2024 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_6_0.Volume",
 "Name": "MyVolume",
 "Id": "1",
 "Description": "Default Volume Description",
 "RAIDType": "RAID1",
 "Identifiers": [
 {
 "DurableNameFormat": "NAA",
 "DurableName": "65456765456761001234076100123487"
 }
],
 "Status": {
 "State": "Enabled",
 "Health": "OK"
 },
 "CapacityBytes": 1099511627776,
}
```

```
 "Links": {
 "StoragePool": "/redfish/v1/Storage/1/StoragePools/1"
 }
 }
```

**Postconditions:** The selected volume is added to the AllocatedVolumes collection within the selected storage pool.

**Failure Scenario:** None defined

**See also:** None defined.

### 5.1.25 *Create Volume specifying Class of Service*

**Summary:** Create a Volume

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

**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:** `Content-type : application/json`
- **Body:**

```
{
 "Name": "Snapshot1",
 "CapacityBytes": 1099511627776,
 "Links": {
```

```
"ClassOfService": {"odata.id":
 ↪ "/redfish/v1/StorageServices/1/ClassesofService/BostonBunker"}
}
}
```

**Response:**

- **HTTP Status:** 201 (Created)
- **Headers:**

Location : /redfish/v1/StorageServices/1/Volumes/3 Content-type  
: application/json

- **Body:**

```
{
 "@SSM.Copyright": "Copyright (c) 2014-2024 SNIA. All
 ↪ rights reserved.",
 "@odata.context":
 ↪ "/redfish/v1/$metadata/#Volume.Volume",
 "@odata.id":
 ↪ "/redfish/v1/StorageServices/1/Volumes/3",
 "@odata.type": "#Volume_1_0_0.Volume",
 "Name": "Snapshot1",
 "Id": "3",
 "Description": "",
 "Identifiers": [
 {
 "DurableNameFormat": "NAA6",
 "DurableName": "65456765456761001234076100123487"
 }
],
 "Manufacturer": "SuperDuperSSD",
 "Model": "Drive Model string",
 "Status": {
 "State": "Enabled",
```

```
"Health": "OK"
 },
 "AccessCapabilities": [
 "Read",
 "Write",
 "Append",
 "Streaming"
],
 "BlockSizeBytes": 512,
 "CapacitySources": [
 {
 "ConsumedBytes": 0,
 "AllocatedBytes": 10737418240,
 "GuaranteedBytes": 536870912,
 "ProvisionedBytes": 1099511627776,
 "Links": {
 "ClassOfService": {
 "@odata.id":
 ↪ "/redfish/v1/StorageServices/1/Links/ClassesOfService/Silver
 },
 "ProvidingPool": {
 "@odata.id":
 ↪ "/redfish/v1/StorageServices/1/StoragePools/SpecialPool"
 }
 }
 }
],
 "CapacityBytes": 1099511627776,
 "Capacity": {
 "Data": {
 "ConsumedBytes": 0,
 "AllocatedBytes": 10737418240,
 "GuaranteedBytes": 1099511627776,
 "ProvisionedBytes": 1099511627776
 },
 "Metadata": {
```



```
 "ConsumedBytes": 536870912,
 "AllocatedBytes": 536870912
 },
 "Links": {
 "ClassofService": {
 "@odata.id":

 ↪ "/redfish/v1/StorageServices/1/Links/ClassesofService/BostonBun
 }
 }
}
```

**Postconditions:** The selected volumes are added to the collection for the Storage Group.

**Failure Scenario:** None defined

**See also:** None defined.

### 5.1.26 Create Volume using Default Class of Service

**Summary:** Create a Volume using the Default Class of Service

**Purpose:** Create a Volume with a known capacity and without specifying the class of service, invoking the DefaultClassOfService.

**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, but does not specify a given class of service. The system will attempt to provision the system with the specified ClassOfService in DefaultClassOfService in either the given StoragePool or the StorageService as available.

**Preconditions:** The DefaultClassOfService property is set in either the StoragePool or StorageService.

**Feature(s):** [Class of Service](#)

**Inputs:**

- URL for Storage Service: /redfish/v1/StorageServices/1
- Requested volume size in bytes (for 1 TiB, here) : 1099511627776
- Requested name of volume: "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

- **Headers:** Content-type : application/json
- **Body:**

```
{
 "Name": "Snapshot1",
 "CapacityBytes": 1099511627776
}
```

**Response:**

- **HTTP Status:** 201 (Created)

- **Headers:**

Location : /redfish/v1/StorageServices/1/Volumes/3  
Content-type : application/json

- **Body:**

```
{
 "@SSM.Copyright": "Copyright (c) 2014-2024 SNIA. All
 ↪ rights reserved.",
 "@odata.context":
 ↪ "/redfish/v1/$metadata/#Volume.Volume",
 "@odata.id": "/redfish/v1/StorageServices/1/Volumes/3",
 "@odata.type": "#Volume_1_0_0.Volume",
 "Name": "Snapshot1",
 "Id": "3",
 "Description": "",
 "Identifiers": [
 {
 "DurableNameFormat": "NAA6",
 "DurableName": "65456765456761001234076100123487"
 }
],
 "Manufacturer": "SuperDuperSSD",
 "Model": "Drive Model string",
 "Status": {
 "State": "Enabled",
 "Health": "OK"
 },
 "AccessCapabilities": [
 "Read",
 "Write",
 "Append",
 "Streaming"
]
}
```

```
],
 "BlockSizeBytes": 512,
 "CapacitySources": [{
 "ConsumedBytes": 0,
 "AllocatedBytes": 10737418240,
 "GuaranteedBytes": 536870912,
 "ProvisionedBytes": 1099511627776,
 "Links": {
 "ClassOfService": {
 "\@odata.id":
 ↪ "/redfish/v1/StorageServices/1/Links/ClassesOfService/SilverBoston"},
 "ProvidingPool": {
 "\@odata.id":
 ↪ "/redfish/v1/StorageServices/1/StoragePools/SpecialPool"
 }
 }
 }],
 "Capacity": {
 "Data": {
 "ConsumedBytes": 0,
 "AllocatedBytes": 10737418240,
 "GuaranteedBytes": 1099511627776,
 "ProvisionedBytes": 1099511627776
 },
 "Metadata": {
 "ConsumedBytes": 536870912,
 "AllocatedBytes": 536870912
 }
 },
 "Links": {
 "ClassofService": {
 "\@odata.id":
 ↪ "/redfish/v1/StorageServices/1/Links/ClassesofService/BostonBunker"
```

```
}
}
}
```

**Postconditions:** The selected volumes are added to the collection for the Storage Group.

**Failure Scenario:** None defined

**See also:** None defined.

### 5.1.27 *Delete an endpoint*

**Summary:** Delete an Endpoint.

**Purpose:** Delete an Endpoint.

**Triggers:** None defined.

**Detailed Context:** Delete an Endpoint.

**Preconditions:** None.

**Feature(s):** [Access Management](#)

**Inputs:**

- Endpoint Name: "InitiatorEndpoint1"

**Basic Course of Events:**

1. Delete the Endpoint from the Endpoints collection.

**Request:**

Delete /redfish/v1/Storage/MidrangeStorageSystem/Endpoints/InitiatorEndpoint1

- **Headers:** Content-type : application/json

- **Body:**

No headers required.

**Response:**

- **HTTP Status:** 200 (OK)

- **Headers:**

Content-type: application/json

- **Body:**

```
{
 "@odata.type": "#Endpoint.v1_7_0.Endpoint",
 "Id": "InitiatorEndpoint1",
 "Name": "InitiatorEndpoint1",
 "Description": "iSCSI target Endpoint 1",
```

```
"Identifiers": [
 {
 "DurableName":
 ↪ "4289a3dd-ba52-41b8-9d3d-38de196e059c",
 "DurableNameFormat": "UUID"
 }
,
 "ConnectedEntities": [
],
 "EndpointProtocol": "iSCSI",
 "IPTransportDetails": [
 {
 "IPv4Address": {
 "Address": "192.168.1.63"
 }
 }
],
 "Status": {
 "State": "Enabled",
 "Health": "OK"
 },
 "@odata.id":
 ↪ "/redfish/v1/Storage/MidrangeStorageSystem/Endpoints/InitiatorEnd
 "@Redfish.Copyright": "Copyright 2015-2024 SNIA. All
 ↪ rights reserved."
}
```

**Postconditions:** None defined.

**Failure Scenario:** None defined

**See also:** None defined.

### 5.1.28 Delete Multiple Drives from an Existing Storage Pool

**Summary:** Delete multiple drives from an Storage Pool.

**Purpose:** Remove multiple drives from a Storage Pool

**Triggers:** Re-assign resources in a pool; this could be performance, capacity or application triggered.

**Detailed Context:** The storage admin needs to decrease the underlying available capacity within an existing pool. This use case makes the following assumptions about the “implementation” servicing the request:

- The implementation manages, or does not have any, constraints on the removal of multiple drives from the target storage pool.

**Preconditions:** User has already selected a Pool, and the Drives to be removed.

**Feature(s):** [Block provisioning](#)

**Inputs:**

- URL for Storage Pool: `/redfish/v1/Storage/1/StoragePools/PrimaryPool`
- Drives to remove: `[{"@odata.id": "/redfish/v1/Chassis/1/Drives/1"}, {"@odata.id": "/redfish/v1/Chassis/1/Drives/2"}]`

**Basic Course of Events:**

1. Use the “RemoveDrives” Action on the `PrimaryPool` storage pool, passing the selected drives as input.

**Request:** `POST /redfish/v1/Storage/1/StoragePools/PrimaryPool.RemoveDrives`

- **Headers:** `Content-type : application/json`

- **Body:**

```
{
 "Drives" : [
 {"@odata.id": "/redfish/v1/Chassis/1/Drives/1"},
 {"@odata.id": "/redfish/v1/Chassis/1/Drives/2"}
]
}
```



**Response:** Response is dependent on implementation's capability.

If the implementation is able to return immediately:

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool
- **Body:** None.

If the implementation requires a background task (using the Redfish task service) to return status:

- **HTTP Status:** 202 (Accepted)
- **Headers:** Location : /redfish/v1/TaskService/Tasks/TaskID2
- **Body:** None.

**Postconditions:** The identified drives have been removed from a capacity source used by the selected storage pool.

**Failure Scenario:** None defined

**See also:** None defined.

### 5.1.29 Delete Volume

**Summary:** Delete an existing Volume

**Purpose:** Delete an existing volume, returning its capacity to the underlying Storage-Pool.

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

**Detailed Context:** The admin needs to repurpose the capacity in a storage pool to satisfy a user or application request.

**Preconditions:** User has already selected a Volume

**Feature(s):** [Block provisioning](#)

**Inputs:**

- URL for the Volume: `/redfish/v1/Storage/1/StoragePools/AllocatedVolumes/1`

**Basic Course of Events:**

1. Delete the Volume from the AllocatedVolumes 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:**

DELETE `/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1`

- **Headers:** Content-type : application/json
- **Body:** None.

**Response:** Response contains the details of the deleted volume.

- **HTTP Status:** 200 (OK)
- **Headers:**  
Content-type : application/json
- **Body:**

```
{
 "@SSM.Copyright": "Copyright (c) 2014-2024 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_6_0.Volume",
 "Name": "MyVolume",
 "Id": "1",
 "Description": "Default Volume Description",
 "RAIDType": "RAID1",
 "Identifiers": [
 {
 "DurableNameFormat": "NAA",
 "DurableName": "65456765456761001234076100123487"
 }
],
 "Status": {
 "State": "Enabled",
 "Health": "OK"
 },
 "CapacityBytes": 1099511627776,
 "Links": {
 "StoragePool": "/redfish/v1/Storage/1/StoragePools/1"
 }
}
```

**Postconditions:** None defined.

**Failure Scenario:** None defined

**See also:** None defined.

### 5.1.30 Deprovision a Namespace

**Summary:** Deprovision a Namespace

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

**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:**

- **HTTP Status:** 204 (No Content)
- **Headers:** No additional headers required.
- **Body:** None defined.

**Postconditions:** The requested namespace has been deprovisioned.

**Failure Scenario:** None defined.

**See also:** [xx](#) for a similar use case that applies to Volumes rather than to Namespaces.

### 5.1.31 *Detach a Namespace*

**Summary:** Detach a Namespace

**Purpose:** Remove visibility to a namespace by detaching it from an IO Controller.

**Triggers:** None defined.

**Detailed Context:** Need to detach a namespace from an IO Controller. The user no longer wants the namespace to be accessible from the host connected to the IO Controller.

**Preconditions:** The IO Controller and Namespace need to exist and be fully defined. The Namespace must be attached to the IO Controller.

**Feature(s):** NVMe

**Inputs:**

- URL for Namespace.
- IO Controller.

**Basic Course of Events:**

1. Get the list of all Attached Namespaces attached to the given IO Controller.

**Request:**

GET /redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Volumes/  
SimpleNamespace

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

**Response:**

- **HTTP Status:** 200 (OK)
- **Headers:**
  - Content-type : application/json
  - ETag : "7765ADC9"
- **Body:**

```
{
 "@Redfish.Copyright": "Copyright 2014-2024 SNIA. All
 ↪ rights reserved.",
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Controllers/NV
 "@odata.type":
 ↪ "#StorageController.v1_0_0.StorageController",
 "Id": "NVMeIOController",
 "Name": "NVMe I/O Controller",
 "Description": "Single NVMe I/O Controller presented
 ↪ to host.",
 "Status": {
 "State": "Enabled",
 "Health": "OK"
 },
 "Manufacturer": "Best NVMe Vendor",
 "Model": "Simple NVMe Device",
 "SerialNumber": "NVME123456",
 "PartNumber": "NVM44",
 "FirmwareVersion": "1.0.0",
 "SupportedControllerProtocols": [
 "PCIe"
],
 "SupportedDeviceProtocols": [
 "NVMe"
],
 "NVMeControllerProperties": {
 "NVMeVersion": "1.3",
 "NVMeControllerAttributes": {
 "ReportsUUIDList": false,
 "SupportsSQAssociations": false,
 "ReportsNamespaceGranularity": false,
 "SupportsTrafficBasedKeepAlive": false,
 "SupportsPredictableLatencyMode": false,
 "SupportsEnduranceGroups": false,
```

```

 "SupportsReadRecoveryLevels": false,
 "SupportsNVMSets": false,
 "SupportsExceedingPowerOfNonOperationalState":
 ↪ false,
 "Supports128BitHostId": false
 }
},
"Links": {
 "AttachedVolumes": [
 {
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Volumes/
 }
]
}
}

```

2. Delete the Attached Namespace from the AttachedVolumes array by PATCHing the IO Controller with an edited set of AttachedVolumes. (Note: This example shows an AttachedVolumes array with only one namespace; therefore the patch command is sent with an empty array.)

**Request:**

PATCH /redfish/v1/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Controller/NVMeIOController/

- **Headers:** Content-type : application/json

- **Body:**

```

{
 "Links": {
 "AttachedVolumes": [
 { }
]
 }
}

```

**Response:**

- **HTTP Status:** 200 (OK)
- **Headers:** Content-type : application/json
- **Body:**

```
{
 "@Redfish.Copyright": "Copyright 2014-2024 SNIA. All
 ↪ rights reserved.",
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Controllers/NV
 "@odata.type":
 ↪ "#StorageController.v1_0_0.StorageController",
 "Id": "NVMeIOController",
 "Name": "NVMe I/O Controller",
 "Description": "Single NVMe I/O Controller presented
 ↪ to host.",
 "Status": {
 "State": "Enabled",
 "Health": "OK"
 },
 "Manufacturer": "Best NVMe Vendor",
 "Model": "Simple NVMe Device",
 "SerialNumber": "NVME123456",
 "PartNumber": "NVM44",
 "FirmwareVersion": "1.0.0",
 "SupportedControllerProtocols": [
 "PCIe"
],
 "SupportedDeviceProtocols": [
 "NVMe"
],
 "NVMeControllerProperties": {
 "NVMeVersion": "1.3",
 "NVMeControllerAttributes": {
 "ReportsUUIDList": false,
```



```
 "SupportsSQAssociations": false,
 "ReportsNamespaceGranularity": false,
 "SupportsTrafficBasedKeepAlive": false,
 "SupportsPredictableLatencyMode": false,
 "SupportsEnduranceGroups": false,
 "SupportsReadRecoveryLevels": false,
 "SupportsNVMSets": false,
 "SupportsExceedingPowerOfNonOperationalState":
 ↪ false,
 "Supports128BitHostId": false
 },
 "Links": {
 "AttachedVolumes": [
]
 }
}
```

**Postconditions:** The Namespace has been removed from the AttachedVolumes array.

**Failure Scenario:** None defined.

**See also:** None defined.

### 5.1.32 *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

**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. Use 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:**

- **HTTP Status:** 200 (OK)
- **Headers:**
  - Content-type : application/json
  - ETag: FD87EC469AE
- **Body:**

```
{
 "@Redfish.Copyright": "Copyright 2015-2024 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",
"Description": "application storage",
"Identifiers": [
{
 "DurableNameFormat": "NAA6",
 "DurableName": "61001234876545676100123487654567"
}
],
"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
},
}
```

```
}
```

2. Use 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:**

- **HTTP Status:** 200 (OK)
- **Headers:**
  - Content-type : application/json
  - ETag: FD87EC469AE
- **Body:**

```
{
"@Redfish.Copyright": "Copyright 2015-2016 SNIA. All
↪ rights reserved.",
"@odata.context":
↪ "/redfish/v1/$metadata#StoragePool.StoragePool",
"@odata.id":
↪ "/redfish/v1/StorageServices/1/StoragePools/BasePool",
"@odata.type": "#StoragePool.1_0_0.StoragePool",
"Id": "BasePool",
"Name": "BasePool",
"Description": "Base storage pool for this storage
↪ service",
"Status": {
"State": "Enabled",
"Health": "OK",
"HealthRollUp": "Degraded"
},
"BlockSizeBytes": 8192,
"Capacity": {
"Data": {
"ConsumedBytes": 824633720832,
```

```
"AllocatedBytes": 1099511627776
},
"Metadata": null,
"Snapshot": null
},
"CapacitySources": [
{
 "ProvidedCapacity": {
 "ConsumedBytes": 70368744177664,
 "AllocatedBytes": 140737488355328,
 "GuaranteedBytes": 17592186044416,
 "ProvisionedBytes": 562949953421312
 },
 "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/610012348765456761001234876
```

```
],
 "SupportedClassesOfService": [
 {
 "@odata.id":
 ↪ "/redfish/v1/StorageServices/1/ClassesOfService/GoldBoston"
 },
 {
 "@odata.id":
 ↪ "/redfish/v1/StorageServices/1/ClassesOfService/SilverBoston"
 }
]
}
```

3. Use expand action to increase size of storage volume by 25%.

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:** Content-type : application/json
- **Body:**

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

**Response:**

- **HTTP Status:** 202 (Accepted)

- **Headers:**

`Location : redfish/v1/TaskService/Tasks//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.



### 5.1.33 *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.

**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:** Content-type : application/json

- **Body:**

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

**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/ConsistencyGroups/CG\_DB1
- **Body:** None.

**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](#)

### 5.1.34 *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.

**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:** Content-type : application/json
- **Body:**

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

**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/Allocated
- **Body:** None.

**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 \(CG\)](#)

### 5.1.35 *Provision a Namespace from NVM Set*

**Summary:** Provision a Namespace from an NVM Set

**Purpose:** Provision a Namespace from an NVMe device that supports Endurance Groups and NVM Sets.

**Triggers:** None defined.

**Detailed Context:** Create a namespace, and allocate resources reserved for it, from an NVMe device that supports Endurance Groups and NVM Sets. The Create Namespace request is performed on the NVM Set 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 appropriate NVM Set and the desired capacity.

**Feature(s):** [NVMe](#)

**Inputs:**

- URL for Storage Pool: `/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/StoragePools/`
- Requested volume size in bytes (1 TiB, here): `1099511627776`
- Requested name of volume: `"MyNamespace"`

**Basic Course of Events:**

1. Post the definition of the new Namespace to the NVM Set resource collection, which is of type `StoragePoolCollection`.

This instructs the service to use the identified NVMSets (StoragePools) to allocate a new Namespace of the requested size. Any additional protection properties will be inherited from the NVM Set.

**Request:**

POST `/redfish/v1/Storage/1/StoragePools/NVMSets`

- **Headers:** `Content-type : application/json`
- **Body:**

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

**Response:**

Response contains the details of the created volume.

- **HTTP Status:** 201 (Created)

- **Headers:**

```
`Location : /redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyNamespace"
Content-type : application/json`
```

- **Body:**

```
{
 "@Redfish.Copyright": "Copyright 2014-2024 SNIA. All
 ↪ rights reserved.",
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyNamespace"
 "@odata.type": "#Volume.v1_9_0.Volume",
 "Id": "1",
 "Name": "MyNamespace",
 "LogicalUnitNumber": 1,
 "Status": {
 "State": "Enabled",
 "Health": "OK"
 },
 "Identifiers": [
 {
 "DurableNameFormat": "NQN",
```

```
 "DurableName":
 ↪ "nqn.2014-08.org.nvmexpress:uuid:6c5fe566-10e6-4fb6-aad4-8b4
 },
],
"Capacity": {
 "Data": {
 "ConsumedBytes": 0,
 "AllocatedBytes": 1099511627776,
 "ProvisionedBytes": 1099511627776
 },
 "CapacitySources": [
 {
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyNa
 }
],
 "NVMeNamespaceProperties": {
 "NamespaceId": "0x224",
 "NamespaceFeatures": {
 "SupportsThinProvisioning": false,
 "SupportsAtomicTransactionSize": false,
 "SupportsDeallocatedOrUnwrittenLBAError": false,
 "SupportsNGUIDReuse": false,
 "SupportsIOPerformanceHints": false
 },
 "LBAFormat": {
 "LBAFormatType": "LBAFormat0",
 "RelativePerformance": "Best",
 "LBADataSizeBytes": 4096,
 "LBAMetadataSizeBytes": 0
 },
 "MetadataTransferredAtEndOfDataLBA": false,
 "NVMeVersion": "1.4"
 }
}
```

}

**Postconditions:** The selected Namespace has been created and is added to the NVM Set.

**Failure Scenario:** None defined.

**See also:** NVMe use case to [Attach a Namespace](#), and to [Provision a Namespace](#) without and NVMSets, and [xx](#) which illustrates Volume deletion, the equivalent of deprovisioning.



### 5.1.36 *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.

**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
- 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:** Content-type : application/json
- **Body:**

```
{
 "Name" : "MyNamespace",
```

```
 "Capacity": {
 "Data": {
 "ProvisionedBytes": 1099511627776
 }
 }
 }
}
```

**Response:**

Response contains the details of the created volume.

- **HTTP Status:** 201 (Created)

- **Headers:**

Location : /redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyName

Content-type : application/json

- **Body:**

```
{
 "@Redfish.Copyright": "Copyright 2014-2024 SNIA.
 ↪ All rights reserved.",
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyNamesp
 "@odata.type": "#Volume.v1_9_0.Volume",
 "Id": "1",
 "Name": "MyNamespace",
 "LogicalUnitNumber": 1,
 "Status": {
 "State": "Enabled",
 "Health": "OK"
 },
 "Identifiers": [
 {
 "DurableNameFormat": "NQN",
 "DurableName":
 ↪ "nqn.2014-08.org.nvmexpress:uuid:6c5fe566-10e6-4fb6-aad4-8
 }
],
}
```

```
"Capacity": {
 "Data": {
 "ConsumedBytes": 0,
 "AllocatedBytes": 1099511627776,
 "ProvisionedBytes": 1099511627776
 }
},
"CapacitySources": [
 {
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyNa
]
},
"NVMeNamespaceProperties": {
 "NamespaceId": "0x224",
 "NamespaceFeatures": {
 "SupportsThinProvisioning": false,
 "SupportsAtomicTransactionSize": false,
 "SupportsDeallocatedOrUnwrittenLBError":
 ↪ false,
 "SupportsNGUIDReuse": false,
 "SupportsIOPerformanceHints": false
 },
 "LBAFormat": {
 "LBAFormatType": "LBAFormat0",
 "RelativePerformance": "Best",
 "LBADataSizeBytes": 4096,
 "LBAMetadataSizeBytes": 0
 },
 "MetadataTransferredAtEndOfDataLBA": false,
 "NVMeVersion": "1.4"
}
```

**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](#), to [Provision a Namespace with NVMSet](#), and [xx](#) for a similar use case that creates a Volume rather than a Namespace.

### 5.1.37 *Provision a Namespace with a specific LBA format*

**Summary:** Provision a Namespace with a specific LBA format

**Purpose:** Provision a Namespace from a simple NVMe device, using a specific LBA format

**Triggers:** None defined.

**Detailed Context:** Create a namespace, and allocate resources reserved for it, using a predefined LBA format type. 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.
- User has already selected the desired LBA format, from the LBA formats supported by the controller (through [Confirm Valid LBA Formats](#))

**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`
- Requested LBAFormatType: `LBAFormat0`
- 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 and LBA format. Only the LBAFormatType property of the LBAFormat is required. Any additional protection properties will be inherited from the NVM Subsystem.

**Request:**

POST `/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes`

- **Headers:** Content-type : application/json

- **Body:**

```
{
 "Name" : "MyNamespace",
 "Capacity": {
 "Data": {
 "AllocatedBytes": 1099511627776,
 "ProvisionedBytes": 1099511627776
 },
 },
 "NVMeNamespaceProperties": {
 "LBAFormat": {
 "LBAFormatType": "LBAFormat0"
 }
 }
}
```

**Response:**

Response contains the details of the created volume.

- **HTTP Status:** 201 (Created)

- **Headers:**

Location : /redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyName

Content-type : application/json

- **Body:**

```
{
 "@Redfish.Copyright": "Copyright 2014-2024 SNIA.
 ↪ All rights reserved.",
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyNamesp
 "@odata.type": "#Volume.v1_9_0.Volume",
 "Id": "1",
 "Name": "MyNamespace",
 "LogicalUnitNumber": 1,
 "Status": {
```

```
 "State": "Enabled",
 "Health": "OK"
 },
 "Identifiers": [
 {
 "DurableNameFormat": "NQN",
 "DurableName":
 ↪ "nqn.2014-08.org.nvmexpress:uuid:6c5fe566-10e6-4fb6-aad4-8
 }
],
 "Capacity": {
 "Data": {
 "ConsumedBytes": 0,
 "AllocatedBytes": 1099511627776,
 "ProvisionedBytes": 1099511627776
 }
 },
 "CapacitySources": [
 {
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/MyNa
 }
],
 "NVMeNamespaceProperties": {
 "NamespaceId": "0x224",
 "NamespaceFeatures": {
 "SupportsThinProvisioning": false,
 "SupportsAtomicTransactionSize": false,
 "SupportsDeallocatedOrUnwrittenLBError":
 ↪ false,
 "SupportsNGUIDReuse": false,
 "SupportsIOPerformanceHints": false
 },
 "LBAFormat": {
 "LBAFormatType": "LBAFormat0",
```

```
 "RelativePerformance": "Best",
 "LBADataSizeBytes": 4096,
 "LBAMetadataSizeBytes": 0
 },
 "MetadataTransferredAtEndOfDataLBA": false,
 "NVMeVersion": "1.4"
}
```

**Postconditions:** The selected Namespace has been created and is added to the NVM Subsystem.

**Failure Scenario:** None defined.

**See also:** NVMe use cases to [Confirm Valid LBA Formats](#), [Attach a Namespace](#), [Provision a Namespace with NVMeSet](#), and [xx](#) which illustrates Volume deletion, the equivalent of deprovisioning.

### 5.1.38 Query Supported Security Protocols

**Summary:** Query the security protocols supported by a controller.

**Purpose:** Discover the security-related capabilities of a device, in order to perform further security-related management.

**Triggers:** Initial device provisioning and enumeration

**Detailed Context:** The storage admin requires a listing of the security protocols supported by a device, in order to communicate with a device using any one of those supported protocols.

A supported protocol will typically be one of:

- 0x00: Security Protocol information
- 0x01 - 0x06: Trusted Computing Group protocols
- 0x40: Security Association Creation capabilities
- 0x41: IKEv2-SCSI
- 0xe9 - 0xea: NVMe Express security protocols

- but arbitrary protocol types are also available, including vendor-defined protocols.



This use-case uses Security Protocol 0 (“Security Protocol Information”) to query the other security protocols implemented on the controller. The Security Send and Security Receive Controller actions provide the transport for this protocol communication.

**Preconditions:** User has selected a storage device to query.

**Features:** [Security Management](#)

**Inputs:**

- URL for Controller: /redfish/v1/Storage/1/Controllers/1
- Secure protocol request parameters: To query supported protocols use:
  - SecurityProtocol (SP): 0
  - SecurityProtocolSpecific (SPSP): 0

**Basic Course of Events:**

1. Invoke the SecurityReceive Action on the Controller, passing the SP and SPSP parameters.

**Request:** POST /redfish/v1/Storage/1/Controllers/1.SecurityReceive

- **Headers:** Content-type : application/json

- **Body:**

```
{
 "SecurityProtocol": 0,
 "SecurityProtocolSpecific": 0
}
```

**Response:**

- **HTTP Status:** 200 (OK)

- **Headers:** None.

- **Body:**

```
{
 "Data": "AAAAAAAAAAIAAQ=="
}
```

2. Decode the base64 "Data" field, and parse the binary response data according to section 5.1.3 of INCITS 501-2016.

In this example, the data returned is:

```
00 00 00 00 00 00 00 02 00 01
```

Which indicates support for two security protocols, 0x00 and 0x01.

**Postconditions:** None.

**Failure Scenario:** None defined.

**See also:** None defined.

### 5.1.39 *Receive Security Protocol Data*

**Summary:** Receive security protocol data from a storage device.

**Purpose:** As part of security-related device management, receive a block of security protocol data from a device which supports that specific security protocol.

**Triggers:** Initial device provisioning and enumeration, device reprovisioning

**Detailed Context:** The storage admin wishes to perform security-management related operations on a storage controller device, through one of the controller's supported security protocols. In order to perform protocol interactions, data may need to be received from the device, using a "Security Receive" operation.

**Preconditions:**

- User has selected a storage device for security management operations
- Device supports the security protocol operations
- Security protocol number is 0, 1 or 2 (respectively, Security Protocol Information, and TCG Storage Architecture protocol 1 and 2)
- Device implements the selected security protocol

**Features:** [Security Management](#)

**Inputs:**

- URL for Controller: `/redfish/v1/Storage/1/Controllers/1`

- Secure protocol request parameters (the following are the required parameters to send data):
  - SecurityProtocol (SP): 2
  - SecurityProtocolSpecific (SPSP): 4100
- Expected response length: 48

### Basic Course of Events:

1. Invoke the `SecurityReceive` Action on the Controller, passing a Security Protocol (SP) parameter, a Security Protocol Specific Parameter (SPSP), and an expected length for response data.

**Request:** POST /redfish/v1/Storage/1/Controllers/1.SecurityReceive

- **Headers:** Content-type : application/json
- **Body:**

```
{
 "SecurityProtocol": 2,
 "SecurityProtocolSpecific": 4100,
 "AllocationLength": 48,
}
```

Security protocol data will be returned in the response as a base64-encoded string.

**Response:**

- **HTTP Status:** 200 (OK)
- **Headers:** None.
- **Body:** json { "Data": "EAQAAAAAAAAEAAAAiAAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" }

**Postconditions:** None.

**Failure Scenario:** None defined.

**See also:** None defined.

#### 5.1.40 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 ConsistencyGroups.

**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 ConsistencyGroup.

**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:** Content-type : application/json
- **Body:**

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

**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/ConsistencyGroups/1
- **Body:** None.

**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](#)

### 5.1.41 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.

**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/65097`

- 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:** Content-type : application/json
- **Body:**

```
{
 "TargetVolume" :
 ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/6509734
 "DeleteTargetVolume" : "false"
}
```

**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/Allocated
- **Body:** None.

**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, as it is 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 \(CG\)](#)

### 5.1.42 Report Namespace Capacity

**Summary:** Report Namespace Capacity

**Purpose:** Report Namespace Capacity

**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 Allocated-Bytes (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/Namesp`

**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/Namesp`

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

**Response:**

- **HTTP Status:** 200 (OK)
- **Headers:**
  - Content-type : `application/json`
  - ETag : `"97AAD42"`



- **Body:**

```
{
 "@Redfish.Copyright": "Copyright 2014-2024 SNIA. All
 ↪ rights reserved.",
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/Namespace1",
 "@odata.type": "#Volume.v1_9_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-aad4-8b4159029
 }
],
 "Capacity": {
 "Data": {
 "ConsumedBytes": 0,
 "AllocatedBytes": 10737418240,
 "ProvisionedBytes": 10737418240
 }
 },
 "CapacitySources": [
 {
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/NVMeSSD-EG/Volumes/Namespace1/C
 }
],
 "NVMeNamespaceProperties": {
```

```
"NamespaceId": "0x224",
"NamespaceFeatures": {
 "SupportsThinProvisioning": false,
 "SupportsAtomicTransactionSize": false,
 "SupportsDeallocatedOrUnwrittenLBError": false,
 "SupportsNGUIDReuse": false,
 "SupportsIOPerformanceHints": false
},
"LBAFormat": {
 "LBAFormatType": "LBAFormat0",
 "RelativePerformance": "Best",
 "LBADataSizeBytes": 4096,
 "LBAMetadataSizeBytes": 0
},
"MetadataTransferredAtEndOfDataLBA": false,
"NVMeVersion": "1.4"
},
"Links": {
}
}
```

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

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

**Failure Scenario:** None defined.

**See also:** None defined.

### 5.1.43 *Report Remaining Life for a Namespace*

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

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

**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 "Predicted-MediaLifeLeftPercent" 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

**Inputs:**

- URL for Namespace

**Basic Course of Events:**

1. GET the Namespace of interest.

**Request:**

```
GET /redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Volumes/
SimpleNamespace
```

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

**Response:**

- **HTTP Status:** 200 (OK)
- **Headers:**
  - Content-type : application/json
  - ETag : "97AAD42"
- **Body:**

```
{
 "@Redfish.Copyright": "Copyright 2014-2024 SNIA.
 ↪ All rights reserved.",
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/SimplestNVMeSSD/Volumes/Simp
 "@odata.type": "#Volume.v1_9_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-
 }
],
 "Capacity": {
 "Data": {
 "ConsumedBytes": 0,
 "AllocatedBytes": 10737418240
 },
 "Metadata": {
 "AllocatedBytes": 536870912
 }
 },
 "NVMeNamespaceProperties": {
 "NamespaceId": "0x22F",
 "NamespaceFeatures": {
 "SupportsThinProvisioning": false,
 "SupportsAtomicTransactionSize": false,
```

```

 "SupportsDeallocatedOrUnwrittenLBAError":
 ↪ false,
 "SupportsNGUIDReuse": false,
 "SupportsIOPerformanceHints": false
 },
 "LBAFormat": {
 "LBAFormatType": "LBAFormat0",
 "RelativePerformance": "Best",
 "LBADataSizeBytes": 4096,
 "LBAMetadataSizeBytes": 0
 },
 "MetadataTransferredAtEndOfDataLBA": false,
 "NVMeVersion": "1.4"
},
"Links": {
 "Drives": {
 "@odata.id":
 ↪ "/redfish/v1/Chassis/SimplestNVMeSSD/Drives/SimplestNVMeSSD"
 }
}
}

```

2. GET the Drive related to the Namespace.

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

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

**Response:**

- **HTTP Status:** 200 (OK)
- **Headers:**
  - Content-type : application/json
  - 'ETag: "97ACD559"'
- **Body:**

```
{
```

```
"@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/Volume
]
 },
 "Actions": {
 "#Drive.Reset": {
```

```
 "target":
 ↪ "/redfish/v1/Chassis/SimplestNVMeSSD/Drives/SimplestNVMeS
 }
}
}
```

**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.

#### 5.1.44 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 ConsistencyGroups, since the temporary condition that led to a suspension of replication has passed.

**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 what 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:** Content-type : application/json

- **Body:**

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



**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/ConsistencyGroups/CG\_DB1
- **Body:** None.

**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 Consistency-Group 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](#)

### 5.1.45 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.

**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:** Content-type : application/json

- **Body:**

```
{
 "TargetVolume" :
 ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/6509734
}
```

**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/Allocated
- **Body:** None.

**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 \(CG\)](#)

#### 5.1.46 *Retrieve latest instance of storage metrics information*

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

**Purpose:**

- Baseline retrieval of performance metrics
- Input to an end-to-end diagnostics workflow to help identify potential bottlenecks

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

**Detailed context:** Diagnostics scripts require retrieval of performance information from multiple layers in the application stack, including infrastructure, to help identify potential bottlenecks during production hours. The most basic retrieval collects Volume-level performance data that can be aggregated and analyzed by a variety of tools. >Note: If information needs to be gathered over time, consider using the telemetry service, as illustrated in [Review Metrics Trends use case](#).

**Preconditions:** None.

**Feature(s):** [Block IO performance](#)

**Inputs:**

- URL of the selected Volume: /Systems/Sys-1/Storage/DirectAttachStorageSystem/Volumes

**Basic course of events:**

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

**Request:** GET /redfish/v1/Systems/Sys-1/Storage/DirectAttachStorageSystem/Volumes

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

**Response:**

- **HTTP Status:** 200 (Success)
- **Headers:**
  - Content-type : application/json
  - ETag : "97AED48652"

- **Body:**

```
{
 "@odata.type": "#VolumeMetrics.v1_1_0.VolumeMetrics",
 "Name": "Volume Metrics",
 "Id": "Metrics",
 "CorrectableIOReadErrorCount": 184,
 "UncorrectableIOReadErrorCount": 0,
 "CorrectableIOWriteErrorCount": 184,
 "UncorrectableIOWriteErrorCount": 0,
 "ConsistencyCheckErrorCount": 0,
 "ConsistencyCheckCount": 10,
 "RebuildErrorCount": 0,
 "StateChangeCount": 20,
 "IOStatistics": {
 "ReadIORequests": 107238423,
 "ReadHitIORequests": 234445346,
 "ReadIOKiBytes": 234456677,
 "ReadIORequestTime": "P18S",
 "WriteIORequests": 98273402234,
 "WriteHitIORequests": 9723598345,
 "WriteIOKiBytes": 2345325,
 "NonIORequests": 23423466,
 "NonIORequestTime": "P24S"
 },
 "Oem": {},
 "@odata.id":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/DirectAttachStorageSystem/Volumes/
 "@Redfish.Copyright": "Copyright 2015-2024 SNIA. All
 ↪ rights reserved."
}
```

**Postconditions:** None defined.

**See also:** - [Metrics White Paper](#) for more information on the telemetry system and metrics definition and retrieval. - [Review Metrics Trends use case](#) for an example of metrics reporting

### 5.1.47 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 ConsistencyGroups, reversing their roles.

**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:** Content-type : application/json
- **Body:**

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

**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/ConsistencyGroups/1
- **Body:** None.

**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](#)

### 5.1.48 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.

**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/AllocatedVo

**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:** Content-type : application/json

- **Body:**

```
{
 "TargetVolume" :
 ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/6
}
```



**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/1
- **Body:** None.

**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 \(CG\)](#)

### 5.1.49 *Review Metrics Trends*

**Summary:** Retrieve capacity utilization data that has been gathered by a previously defined MetricReportDefinition.

**Purpose:**

- Review recent data that has been retrieved and aggregated by the telemetry service

**Detailed context:** Once appropriate metrics and reports have been defined in the Telemetry service, detailed metrics (potentially including automatic calculations and aggregation) can be retrieved easily.

**Preconditions:**

- Telemetry service has been implemented and enabled.
- Appropriate metrics and metrics reports have been defined.

**Feature(s):** [Block IO performance](#)

**Inputs:**

- URL of the selected MetricsReport: TelemetryService/MetricReports/CapacityUtilization

**Basic course of events:**

1. uses GET request to collect the appropriate report information.

**Request:** GET /redfish/v1/TelemetryService/MetricReports/CapacityUtilization

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

**Response:**

- **HTTP Status:** 200 (Success)
- **Headers:**
  - Content-type : application/json
  - ETag : "97AED48652"
- **Body:**

```
{
 "@odata.type": "#MetricReport.v1_4_2.MetricReport",
 "Id": "CapacityUtilization",
 "Name": "Capacity Utilization Performance Metric
 ↪ Report",
 "ReportSequence": "15",
 "MetricReportDefinition": {
 "@odata.id":
 ↪ "/redfish/v1/TelemetryService/MetricReportDefinitions/CapacityUtili
 },
 "MetricValues": [
 {
 "MetricId": "CapacityUtilization",
 "MetricValue": "99",
 "Timestamp": "2021-04-29T12:25:00-05:00",
 "MetricProperty":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/DirectAttachStorageSystem/
 },
 {
 "MetricId": "CapacityUtilization",
 "MetricValue": "98",
 "Timestamp": "2021-04-29T13:25:00-05:00",
 "MetricProperty":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/DirectAttachStorageSystem/
 },
 {
 "MetricId": "CapacityUtilization",
 "MetricValue": "97",
 "Timestamp": "2021-04-29T14:25:00-05:00",
 "MetricProperty":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/DirectAttachStorageSystem/
 },
 {
 "MetricId": "CapacityUtilization",
 "MetricValue": "96",
```

```

 "Timestamp": "2021-04-29T15:25:00-05:00",
 "MetricProperty":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/DirectAttachStorageSystem/
 },
 {
 "MetricId": "CapacityUtilization",
 "MetricValue": "95",
 "Timestamp": "2021-04-29T16:25:00-05:00",
 "MetricProperty":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/DirectAttachStorageSystem/
 },
 {
 "MetricId": "CapacityUtilization",
 "MetricValue": "94",
 "Timestamp": "2021-04-29T17:25:00-05:00",
 "MetricProperty":
 ↪ "/redfish/v1/Systems/Sys-1/Storage/DirectAttachStorageSystem/
 }
],
"@odata.id":
 ↪ "/redfish/v1/TelemetryService/MetricReports/CapacityUtilization",
"@Redfish.Copyright": "Copyright 2015-2024 SNIA. All
 ↪ rights reserved."
}

```

**Postconditions:** None defined.

**See also:** - [Metrics White Paper](#) for more information on the telemetry system and metrics definition and retrieval. - [IO Performance Metrics use case](#) for an example of metrics retrieval

### 5.1.50 *Send Security Protocol Data*

**Summary:** Send security protocol data to a storage device.

**Purpose:** As part of security-related device management, send a block of security protocol data to a device that supports that specific security protocol.

**Triggers:** Initial device provisioning and enumeration, device reprovisioning

**Detailed Context:** The storage admin wishes to perform security-management related operations on a storage controller device, through one of the controller's supported security protocols. In order to perform protocol interactions, data may need to be transmitted to the device, using a "Security Send" operation.

**Preconditions:**

- User has selected a storage device for security management operations
- Device supports the security protocol operations
- Security protocol number is 1 or 2 (the TCG Storage Architecture protocols supported by Swordfish)
- Device implements the selected security protocol

**Features:** [Security Management](#)

**Inputs:**

- URL for Controller: `/redfish/v1/Storage/1/Controllers/1`
- Secure protocol request parameters (these are the settings required to send data):
  - SecurityProtocol (SP): 2
  - SecurityProtocolSpecific (SPSP): 4100
- Security protocol data: `10 04 00 00 00 00 00 01`, hex bytes.

**Basic Course of Events:**

1. Invoke the SecuritySend Action on the Controller, passing a Security Protocol (SP) parameter, and a Security Protocol Specific Parameter (SPSP), and arbitrary protocol data, as a base64-encoded string.

**Request:** `POST /redfish/v1/Storage/1/Controllers/1.SecuritySend`

- **Headers:** `Content-type : application/json`
- **Body:**

```
{
 "SecurityProtocol": 2,
 "SecurityProtocolSpecific": 4100,
 "Data": "EAQAAAAAAAE="
}
```

**Response:**

- **HTTP Status:** 201 (No Content)
- **Headers:** None.
- **Body:** None.

**Postconditions:** None.

**Failure Scenario:** None defined.

**See also:** None defined.

### 5.1.51 *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.

**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:** Content-type : application/json

- **Body:**

```
{
 "TargetVolume":
 ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/6
}
```

**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/Allocated
- **Body:** None.

**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 \(CG\)](#)



### 5.1.52 *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.

**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:** Content-type : application/json.

- **Body:**

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

**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/ConsistencyGroups/1
- **Body:** None.

**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](#)

### 5.1.53 *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.

**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:

```
["LowSpaceWarningThresholdTriggered",
↪ "LowSpaceWarningThresholdCleared"]
```

**Basic course of events:**

1. Retrieve the event destination

**Request:**

POST /redfish/v1/EventService/Subscriptions/Members

- **Headers:** Content-type : application/json
- **Body:**

```
{
 "@Redfish.Copyright": "Copyright 2016-2024 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:**

- **HTTP Status:** 201 (Created)
- **Headers:**

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.

#### 5.1.54 *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.

**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:** Content-type : application/json

- **Body:**

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

**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/ConsistencyGroups/CG\_DB1

- **Body:** None.

**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](#)

### 5.1.55 *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.

**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/AllocatedVo

**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:** Content-type : application/json

- **Body:**

```
{
 "TargetVolume":
 ↪ "/redfish/v1/Storage/1/StoragePools/PrimaryPool/AllocatedVolumes/6
}
```



**Response:**

- **HTTP Status:** 204 (No Content)
- **Headers:** Location : /redfish/v1/Storage/1/StoragePools/PrimaryPool/Allocated
- **Body:** None.

**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 \(CG\)](#)

### 5.1.56 *Update access rights on an existing volume*

**Summary:** Update existing Connection information to make an existing Volume read-only.

**Purpose:** Remove write permission from an existing Volume, by updating Connection information.

**Triggers:** None defined.

**Detailed Context:** Modify the AccessCapabilities entry within the Connections for a given Volume, removing write permissions, and making the Volume read-only through that Connection.

**Preconditions:** User has already identified the Endpoints from which write permission will be removed.

**Feature(s):** [NVMe](#)

**Inputs:**

None.

**Basic Course of Events:**

1. Retrieve the current set of Connections

**Request:**

GET /redfish/v1/Systems/Sys-1/Fabrics/NVMeoF/Connections

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

**Response:**

- **HTTP Status:** 200 (OK)
- **Headers:**  
Content-type : application/json 'ETag: "97ACD559"'
- **Body:**

```
{
 "@odata.type":
 ↪ "#ConnectionCollection.ConnectionCollection",
 "Name": "NVMeoF Connection Collection",
 "Members@odata.count": 2,
 "Members": [
 {
 "@odata.id":
 ↪ "/redfish/v1/Fabrics/NVMeoF/Connections/1"
 },
 {
 "@odata.id":
 ↪ "/redfish/v1/Fabrics/NVMeoF/Connections/3"
 }
],
 "@odata.id": "/redfish/v1/Fabrics/NVMeoF/Connections",
 "@Redfish.Copyright": "Copyright 2015-2021 SNIA. All rights
 ↪ reserved."
}
```

2. Retrieve individual Connections, filtering on the selected Initiator to locate the one to update.

**Request:**

GET /redfish/v1/Systems/Sys-1/Fabrics/NVMeoF/Connections/1

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

- **Body:**

None.

**Response:**

- **HTTP Status:** 200 (OK)

- **Headers:**

Content-type : application/json 'ETag: "11BBC933"

- **Body:**

```
{
 "@odata.type": "#Connection.v1_0_0.Connection",
 "Id": "1",
 "Name": "Connection info for host 1",
 "ConnectionType": "Storage",
 "VolumeInfo": [
 {
 "AccessCapabilities": [
 "Read",
 "Write"
],
 "Volume": {
 "@odata.id":
 ↪ "/redfish/v1/Storage/IPAttachedDrive1/Volumes/SimpleNamespac
 }
 },
 {
 "AccessCapabilities": [
 "Read",
 "Write"
],
 "Volume": {
 "@odata.id":
 ↪ "/redfish/v1/Storage/IPAttachedDrive2/Volumes/SimpleNamespac
 }
 }
],
 "Links": {
 "InitiatorEndpoints": [
 {
 "@odata.id":
 ↪ "/redfish/v1/Fabrics/NVMeoF/Endpoints/Initiator1"
 }
],
 "TargetEndpointGroups": [
```

```

 {
 "@odata.id":
 ↪ "/redfish/v1/Fabrics/NVMeoF/EndpointGroups/TargetEPs"
 }
],
 "@odata.id": "/redfish/v1/Fabrics/NVMeoF/Connections/1",
 "@Redfish.Copyright": "Copyright 2015-2021 SNIA. All rights
 ↪ reserved."
}

```

3. Update the VolumeInfo entry to remove write permission on the Volume through this Connection.

**Request:**

PATCH /redfish/v1/Systems/Sys-1/Fabrics/NVMeoF/Connections/1/VolumeInfo

- **Headers:** Content-type : application/json. 'ETag: "11BBC933"
- **Body:**

```

{
 "AccessCapabilities": [
 "Read"
],
 "Volume": {
 "@odata.id":
 ↪ "/redfish/v1/Storage/IPAttachedDrive2/Volumes/SimpleNamespac
 }
}

```

**Response:**

- **HTTP Status:** 200 (OK)
- **Headers:**

Content-type : application/json 'ETag: "11BBD004"
- **Body:**

```
{
 "@odata.type": "#Connection.v1_0_0.Connection",
 "Id": "1",
 "Name": "Connection info for host 1",
 "ConnectionType": "Storage",
 "VolumeInfo": [
 {
 "AccessCapabilities": [
 "Read",
 "Write"
],
 "Volume": {
 "@odata.id":
 ↪ "/redfish/v1/Storage/IPAttachedDrive1/Volumes/SimpleNames
 }
 },
 {
 "AccessCapabilities": [
 "Read"
],
 "Volume": {
 "@odata.id":
 ↪ "/redfish/v1/Storage/IPAttachedDrive2/Volumes/SimpleNames
 }
 }
],
 "Links": {
 "InitiatorEndpoints": [
 {
 "@odata.id":
 ↪ "/redfish/v1/Fabrics/NVMeoF/Endpoints/Initiator1"
 }
],
 "TargetEndpointGroups": [
 {
```

```
 "@odata.id":
 ↪ "/redfish/v1/Fabrics/NVMeoF/EndpointGroups/TargetEPs"
 }
]
,
"@odata.id": "/redfish/v1/Fabrics/NVMeoF/Connections/1",
"@Redfish.Copyright": "Copyright 2015-2021 SNIA. All
 ↪ rights reserved."
}
```

**Postconditions:** Write permission has been removed for accesses to the selected volume and the given Connection.

**Failure Scenario:** None defined.

**See also:** None defined.

### 5.1.57 Use Features Registry to confirm functionality

**Summary:** Search features registry to confirm that a particular feature is supported by the implementation.

**Purpose:** Need to confirm support for particular features before enabling related client functionality.

**Triggers:** None.

**Detailed Context:** The client interface needs to confirm support for replication before enabling the related UI elements.

**Preconditions:** None.

**Feature(s):** None

**Inputs:** None

#### Basic Course of Events:

1. Locate the features registry for the instance

**Request:** GET /redfish/v1/Registries

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

#### Response:

- **HTTP Status:** 200 (OK)
- **Headers:** ETag : "97A36E3"
- **Body:**

```
{
"@Redfish.Copyright": "Copyright 2015-2024 SNIA. All rights
↪ reserved.",
"@odata.id": "/redfish/v1/Registries",
"@odata.type":
↪ "#MessageRegistryFileCollection.MessageRegistryFileCollection",
"Name": "Registry File Collection",
"Description": "Registry Repository",
```



```
"Members@odata.count": 1,
"Members": [
 {
 "@odata.id":
 ↪ "/redfish/v1/Registries/AdvertisedFeatures.v1_0_0"
 }
]
```

2. Use the appropriate entry to retrieve the features registry locations for Swordfish

**Request:** GET /redfish/v1/Registries/AdvertisedFeatures.v1\_0\_0

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

**Response:**

- **HTTP Status:** 200 (OK)
- **Headers:** ETag : "97A4498D"
- **Body:**

```
{
 "@Redfish.Copyright": "Copyright 2015-2024 SNIA. All
 ↪ rights reserved.",
 "@odata.id":
 ↪ "/redfish/v1/Registries/AdvertisedFeatures.v1_0_0",
 "@odata.type":
 ↪ "#MessageRegistryFile.v1_1_3.MessageRegistryFile",
 "Id": "AdvertisedFeatures.v1_0_0",
 "Name": "Swordfish Feature Registry File",
 "Description": "Swordfish Feature Registry File
 ↪ locations",
 "Registry": "SwordfishFeaturesRegistry.1.2.1",
 "Location": [
 {
 "Language": "en",
```

```
 "PublicationUri":
 ↪ "https://contoso.com/productX/featureinfo/AdvertisedFeatures.v
 "Uri":
 ↪ "/redfish/v1/Registries/AdvertisedFeatures.v1_0_0.json"
 }
],
 "Oem": {}
}
```

3. Use the URI property in location to retrieve the list of supported Features, and confirm the presence of replication.

**Request:** GET /redfish/v1/Registries/AdvertisedFeatures.v1\_0\_0.json

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

**Response:**

- **HTTP Status:** 200 (OK)
- **Headers:** ETag : "1079DD4"
- **Body:**

```
{
 "@odata.type":
 ↪ "#FeaturesRegistry.v1_1_0.FeaturesRegistry",
 "Id": "AdvertisedFeatures.v1_0_0",
 "Name": "Global Swordfish Features Registry",
 "Language": "en",
 "RegistryPrefix": "SwordfishFeatureRegistry",
 "RegistryVersion": "1.2.0",
 "OwningEntity": "SNIA",
 "Features": [
 {
 "FeatureName": "SNIA.Swordfish.Discovery",
 "Description": "Supports discovery of resources in a
 ↪ Swordfish system.",
```

```
"Version": "1.1.0",
"CorrespondingProfileDefinition":
 ↪ "SwordfishDiscovery.json"
},
{
 "FeatureName": "SNIA.Swordfish.Block.Provisioning",
 "Description": "Supports the Block Provisioning
 ↪ Feature.",
 "Version": "1.1.0",
 "CorrespondingProfileDefinition":
 ↪ "SwordfishBlockProvisioning.json"
},
{
 "FeatureName":
 ↪ "SNIA.Swordfish.Block.LocalReplication",
 "Description": "Supports the Local Block Replication
 ↪ Feature.",
 "Version": "1.1.0",
 "CorrespondingProfileDefinition":
 ↪ "SwordfishBlockProvisioning.json"
}
],
"@Redfish.Copyright": "Copyright 2015-2024 SNIA. All
 ↪ rights reserved."
}
```

**Postconditions:** None defined.

**Failure Scenario:** None defined

**See also:** None defined.