



Metrics and Telemetry in Swordfish

Version: 1.2.7

Abstract: This paper defines the approach, infrastructure and mechanisms to use for Swordfish implementations, to capture and store historical metrics to present to Swordfish clients in a standardized fashion, using the Redfish telemetry service.

Working Draft

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

Last Updated: 21 May 2024

Contents

USAGE 3

DISCLAIMER 4

Current Revision 4

Contact SNIA 4

FEEDBACK AND INTERPRETATIONS 5

INTENDED AUDIENCE 5

VERSIONING POLICY 5

Revision History 6

About SNIA 6

Acknowledgements 6

Executive Summary 7

1 Management Demands Metrics 8

2 Telemetry Model 9

2.1 Overview 9

2.2 MetricDefinitions 9

2.3 MetricReportDefinitions 13

2.4 Metric Reports 16

3 Metrics and Telemetry Within Swordfish 20

3.1 Mechanisms for Resource Identification 20

3.2 Value-add Customization 21

For More Information 22

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
12 April 2022	1.2.4	Initial version; Release as Working Draft
12 July 2022	1.2.4a	Release as SNIA Standard.
16 March 2023	1.2.5	Release as Working Draft
20 June 2023	1.2.5a	Release as SNIA Standard
22 January 2024	1.2.6	Release as Working Draft
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, start-ups, 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 white paper, would like to recognize the significant contributions made by the following members listed in Table 2.

Table 2: Contributors

Member	Representatives (* – prior employer)
Intel Corporation	Richelle Ahlvers

Executive Summary

This paper defines the approach, infrastructure and mechanisms to use for Swordfish implementations, to capture and store historical metrics to present to Swordfish clients in a standardized fashion, using the Redfish telemetry service.

Using capacity metrics use cases, we will highlight the expected standard usage of the telemetry service to enable standard client interactions across implementations. We will also provide guidance for expected base functionality, as well as describing how the implementations can provide additional value-add capability accessible to the clients.

1 Management Demands Metrics

Storage management clients rely on detailed, historical information in several areas of the overall system to watch for issues, perform diagnostics and system reconfigurations, provide data for billing and usage forecasting, and other actions.

Some storage-specific areas of interest are performance analysis and capacity utilization trend analysis. Analysis of both of these areas generally requires ongoing data trend analysis, provided over time by historical data logging of specific attributes and values. Note: While systems can provide a single “point-in-time” representation of these attributes, we are referring here to the need for more advanced data representation and analysis. The mechanism to provide collection, storage, and presentation of such information within the Redfish and Swordfish ecosystem is the telemetry service.

2 Telemetry Model

2.1 Overview

The Redfish telemetry service provides the instantiation of the Redfish telemetry model. It includes multiple components; we will focus on the metrics (MetricDefinitions, MetricReportDefinitions, and MetricReports) for the purposes of this paper.

The interactions of interest, and basic dependencies, in the telemetry service with properties in Redfish/Swordfish objects are shown in Figure 1.

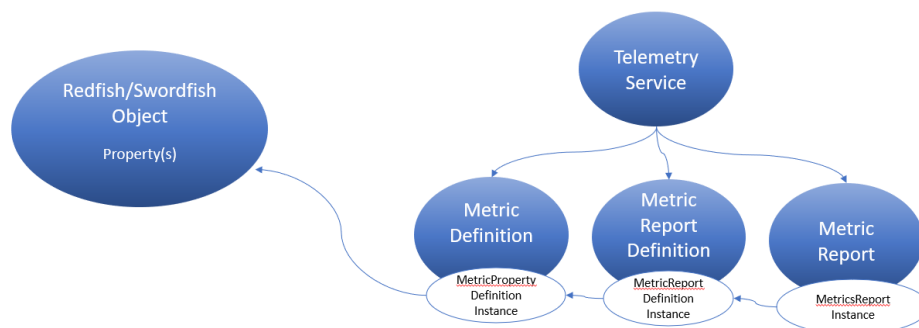


Figure 1: Using the telemetry service to manage metrics

For a more detailed overview of the Redfish telemetry model, you can review the “Redfish Telemetry White Paper” (DSP2051). It will provide a general overview of the model, as well as the properties of each component in the model. [https://www.dmtf.org/sites/default/files/standards/documents/DSP2051_1.0.0.pdf]

2.2 MetricDefinitions

The data retrieved from a Redfish/Swordfish object is discrete, point in time, observations. A metric is defined as a time series, trend of data, with the capability of combining multiple discrete points of data into a richer set of available information.

For a property (or set of properties) of interest, there will be a corresponding MetricDefinition. There may be more than one MetricDefinition that points to a given property or set of properties, using different calculations or analyses.

The MetricDefinition provides context around the properties that metrics refer to, building a framework for developing any analysis or context for the data. A Met-

`MetricDefinition` contains references to the properties it relates to, as well as information about the appropriate type and description of calculations for these properties, any data manipulation required (e.g., data offset, calibration), the data type, min and max values, and additional similar descriptive attributes. It also contains pointers to any, or every, occurrence of the property or set of properties in the system.

The `MetricDefinition` can be thought of having two sections:

- Header - contains all of the supporting property definitions
- Location - contains the `Wildcards` and `MetricProperties` definitions, which provide information about properties and their specific referenced locations in the system

The following `MetricDefinition` provides a sample usage for a Capacity Metric.

Sample `MetricDefinition`:

The set of properties defined in the metric definition itself includes only those properties that describe the metric. For simple metrics such as the capacity metric in this example, this simply includes a small set of properties describing the metric type, units, and context.

```
{
  "@odata.type":
    ↪ "#MetricDefinition.v1_1_0.MetricDefinition",
  "Id": "AllocatedBytes",
  "Name": "Allocated Bytes Metric Definition",
  "MetricType": "Numeric",
  "MetricDataType": "Integer",
  "LogicalContexts": ["Capacity", "Storage"],
  "Units": "Bytes"
}
```

The wildcards section has multiple purposes. This usage reflects the base implementation requirements. Refer to the value-add requirements section for additional usage capabilities. Using the `Name` property (only) in conjunction with the `MetricProperties` section below defines the scope of the metric definition to apply to all instances of the metric within the Swordfish model.

```
{
  "Wildcards": [{
    "Name": "SystemID"
  }],
}
```

```

    {
      "Name": "StorageSystemID"
    },
    {
      "Name": "SystemStoragePoolID"
    },
    {
      "Name": "SystemVolumeID"
    },
    {
      "Name": "StorageID"
    },
    {
      "Name": "StoragePoolID"
    },
    {
      "Name": "VolumeID"
    },
    {
      "Name": "FileSystemID"
    }
  ]
}

```

The `MetricProperties` section contains URI pointers to all potential locations in the system. The wildcards above are uniquely used in conjunction with these URI constructs to identify all of the potential locations for properties to which this metric type can apply, throughout the Redfish and Swordfish schema.

As noted in this example, the Swordfish templates will generally include all possible metric value locations as part of the base functionality. Implementations may choose to implement additional capabilities within the constraints already provided in the telemetry service to allow clients to restrict the application of the metric to subsets of the system.

```

{
  "MetricProperties": [
    "/redfish/v1/Systems/{SystemID}/Storage/{SystemStorageID}/StoragePools/

```

```

    {SystemStorage-
    ↪ PoolID}/AllocatedVolumes/{SystemVolumeID}/Capacity#/
      Data/AllocatedBytes",
    "/red-
    ↪ fish/v1/Storage/{StorageID}/StoragePools/{StoragePoolID}/
      AllocatedVol-
    ↪ umes/{VolumeID}/Capacity#/Data/AllocatedBytes",
    "/red-
    ↪ fish/v1/StorageServices/{StorageServiceID}/StoragePools/
      {ServiceStorage-
    ↪ PoolID}/AllocatedVolumes/{ServiceVolumeID}/
      Capacity#/Data/AllocatedBytes",
    "/red-
    ↪ fish/v1/Storage/{StorageID}/FileSystems/{FileSystemID}/
      Capacity#/Data/AllocatedBytes",
    "/red-
    ↪ fish/v1/Systems/{SystemID}/Storage/{SystemStorageID}/
      FileSys-
    ↪ tems/{SystemFileSystemID}/Capacity#/Data/AllocatedBytes",
    "/red-
    ↪ fish/v1/StorageServices/{StorageServiceID}/FileSystems/
      {ServiceFileSystemID}/Capacity#/Data/AllocatedBytes",
    "/red-
    ↪ fish/v1/Storage/{StorageID}/FileSystems/{FileSystemID}/
      RemainingCapacity#/Data/AllocatedBytes",
    "/red-
    ↪ fish/v1/Systems/{SystemID}/Storage/{SystemStorageID}/
      FileSys-
    ↪ tems/{SystemFileSystemID}/RemainingCapacity#/Data/AllocatedBytes",
    "/red-
    ↪ fish/v1/StorageServices/{StorageServiceID}/FileSystems/
      {ServiceFileSys-
    ↪ temID}/RemainingCapacity#/Data/AllocatedBytes"
  ],
  "@odata.id":
    ↪ "/redfish/v1/TelemetryService/MetricDefinitions/
      AllocatedBytes",

```

```
"@Redfish.Copyright": "Copyright 2014-2022 SNIA. All  
  ↪ rights reserved."  
}
```

2.3 MetricReportDefinitions

A `MetricReportDefinition` builds on a `MetricDefinition`, providing information and additional constraints in order to generate the metric reports themselves. These include specifying the frequency (and type) of data collection, metric calculation properties, information on, and reporting mechanisms with properties including metric logs specific to the metric. These can also include additional scope constraints above those specified in the metric definition.

The `MetricReportDefinition` can also be thought of having two sections:

- Header - contains all of the supporting property definitions
- Location - contains the `Wildcards` and `MetricProperties` definitions, which provide additional information and context about specific referenced locations in the system

The following `MetricReportDefinition` provides a sample usage for a capacity metric report.

The properties included in the `MetricReportDefinition` specify the scheduling parameters for gathering the metrics, including frequency and periodicity. They also include properties to specify what to do with the output: e.g., send as events, or send to logs. If logged, it includes where to log, and additional properties regarding log management.

```
{  
  "@odata.type": "#MetricReportDefini-  
    ↪ tion.v1_3_0.MetricReportDefinition",  
  "Id": "TotalAllocatedCapacity",  
  "Name": "Total Allocated Capacity",  
  "MetricReportDefinitionType": "Periodic",  
  "MetricReportDefinitionEnabled": true,  
  "Schedule": {  
    "RecurrenceInterval": "PT1H"  
  },  
  "ReportActions": [  
    "LogToMetricReportsCollection"  
  ]  
}
```

```
],
  "ReportUpdates": "NewReport",
  "MetricReport": {
    "@odata.id":
      ↪ "/redfish/v1/TelemetryService/MetricReports/
        TotalAllocatedCapacity"
  },
  "Status": {
    "State": "Enabled"
  }
}
```

As with the underlying metric definitions, the metric report definitions also include information about the scope of the system to target.

For the Swordfish templates, this scope will largely be identical to that of the underlying metric, covering the entire system. Implementations may choose to support customization here as well, both within individual metric reports, as well as supporting multiple metric reports against the same metrics with different scope, time intervals, report actions (destinations), or other properties varied.

```
{
  "Wildcards": [{
    "Name": "SystemID"
  },
  {
    "Name": "StorageSystemID"
  },
  {
    "Name": "SystemStoragePoolID"
  },
  {
    "Name": "SystemVolumeID"
  },
  {
    "Name": "StorageID"
  },
  {
    "Name": "StoragePoolID"
  },
  },
}
```

```

    {
      "Name": "VolumeID"
    },
    {
      "Name": "FileSystemID"
    }
  ]
}

```

The `Metrics` section of the report definition includes both pointers to the metrics in scope, as well as information about the data collection that needs to be done on each report generation cycle.

In this example, the data collection will be done once an hour, and the value at that point is of interest (e.g., no calculation of multiple sampled data across the hour is necessary; a simple data point collection will suffice).

```

{
  "Metrics": [
    {
      "MetricId": "AllocatedCapacity",
      "MetricProperties": [
        "/red-
        ↪ fish/v1/Systems/{SystemID}/Storage/{SystemStorageID}/
        ↪ StoragePools/{SystemStoragePoolID}/AllocatedVolumes/
        ↪ {SystemVolumeID}/Capacity#/Data/AllocatedBytes",
        "/red-
        ↪ fish/v1/Storage/{StorageID}/StoragePools/{StoragePoolID}/
        ↪ AllocatedVolumes/{VolumeID}/Capacity#/Data/AllocatedBytes",
        "/red-
        ↪ fish/v1/StorageServices/{StorageServiceID}/StoragePools/
        ↪ {ServiceStoragePoolID}/AllocatedVolumes/{ServiceVolumeID}/
        ↪ Capacity#/Data/AllocatedBytes",
        "/red-
        ↪ fish/v1/Storage/{StorageID}/FileSystems/{FileSystemID}/
        ↪ Capacity#/Data/AllocatedBytes",

```

```

        "/red-
        ↪ fish/v1/Systems/{SystemID}/Storage/{SystemStorageID}

    ↪ /FileSystems/{SystemFileSystemID}/Capacity#/Data/
        AllocatedBytes",
        "/red-
        ↪ fish/v1/StorageServices/{StorageServiceID}/FileSystems
        /{ServiceFileSys-
    ↪ temID}/Capacity#/Data/AllocatedBytes",
        "/red-
        ↪ fish/v1/Storage/{StorageID}/FileSystems/{FileSystemID}
        /RemainingCapacity#/Data/AllocatedBytes",
        "/red-
        ↪ fish/v1/Systems/{SystemID}/Storage/{SystemStorageID}/
        FileSys-
    ↪ tems/{SystemFileSystemID}/RemainingCapacity#/Data/
        AllocatedBytes",
        "/red-
        ↪ fish/v1/StorageServices/{StorageServiceID}/FileSystems/
        {ServiceFileSys-
    ↪ temID}/RemainingCapacity#/Data/AllocatedBytes"
    ],
    "CollectionFunction": "Maximum",
    "CollectionTimeScope": "Point",
    "CollectionDuration": "PT1H"
  }
],
"@odata.id": "/red-
    ↪ fish/v1/TelemetryService/MetricReportDefinitions/
    TotalAllocatedCapacity",
"@Redfish.Copyright": "Copyright 2014-2022 SNIA. All
    ↪ rights reserved."
}

```

2.4 Metric Reports

MetricReports are the output: they contain the output records, as specified in the corresponding MetricReportDefinition. These can be from a single run,

or a series over time, depending on the properties set in the `MetricReportDefinition`.

Metrics can also be set to be sent via Redfish Events; reporting via events is not covered in detail in this white paper.

The following `MetricReport` shows a sample output for a Capacity Metric.

```
{
  "@odata.type": "#MetricReport.v1_4_2.MetricReport",
  "Id": "TotalAllocatedCapacity",
  "Name": "Total Allocated Capacity Performance Metric
  ↪ Report",
  "ReportSequence": "18",
  "MetricReportDefinition": {
    "@odata.id": "/red-
    ↪ fish/v1/TelemetryService/MetricReportDefinitions/
      TotalAllocatedCapacity"
  },
  "MetricValues": [
    {
      "MetricId": "TotalAllocatedCapacity",
      "MetricValue": "300067890136",
      "Timestamp": "2021-04-29T12:25:00-05:00",
      "MetricProperty":
        ↪ "/redfish/v1/Systems/Sys-1/Storage/
          DirectAttachStorageSys-
          ↪ tem/Volumes/Volume1/Capacity#/Data/
            AllocatedBytes"
    },
    {
      "MetricId": "TotalAllocatedCapacity",
      "MetricValue": "300067890136",
      "Timestamp": "2021-04-29T12:25:00-05:00",
      "MetricProperty":
        ↪ "/redfish/v1/Systems/Sys-1/Storage/
          DirectAttachStorageSys-
          ↪ tem/Volumes/Volume1/Capacity#/Data/
            AllocatedBytes"
    },
  ],
}
```

```
{
  "MetricId": "TotalAllocatedCapacity",
  "MetricValue": "300067890136",
  "Timestamp": "2021-04-29T12:25:00-05:00",
  "MetricProperty":
    ↪  "/redfish/v1/Systems/Sys-1/Storage/
      DirectAttachStorageSys-
↪ tem/Volumes/Volume1/Capacity#/Data/
      AllocatedBytes"
},
{
  "MetricId": "TotalAllocatedCapacity",
  "MetricValue": "300067890136",
  "Timestamp": "2021-04-29T12:30:00-05:00",
  "MetricProperty":
    ↪  "/redfish/v1/Systems/Sys-1/Storage/
      DirectAttachStorageSys-
↪ tem/Volumes/Volume1/Capacity#/Data/
      AllocatedBytes"
},
{
  "MetricId": "TotalAllocatedCapacity",
  "MetricValue": "300067890136",
  "Timestamp": "2021-04-29T12:35:00-05:00",
  "MetricProperty":
    ↪  "/redfish/v1/Systems/Sys-1/Storage/
      DirectAttachStorageSys-
↪ tem/Volumes/Volume1/Capacity#/Data/
      AllocatedBytes"
},
{
  "MetricId": "TotalAllocatedCapacity",
  "MetricValue": "300067890136",
  "Timestamp": "2021-04-29T12:40:00-05:00",
  "MetricProperty":
    ↪  "/redfish/v1/Systems/Sys-1/Storage/
      DirectAttachStorageSys-
↪ tem/Volumes/Volume1/Capacity#/Data/
```

```
        AllocatedBytes"
    }
],
"@odata.id":
    ↪ "/redfish/v1/TelemetryService/MetricReports/
    TotalAllocatedCapacity",
"@Redfish.Copyright": "Copyright 2014-2022 SNIA. All
    ↪ rights reserved."
}
```

3 Metrics and Telemetry Within Swordfish

3.1 Mechanisms for Resource Identification

There are three different permutations for determining the scope of applicability of the metrics or metrics definition.

1. Provide references to locations using the `Wildcards` property in conjunction with the URIs specified in the Metrics (`MetricProperties` or `Metric` fields) using no explicitly named instances; this definition means that all instances in each location are in scope.

This mechanism is provided out-of-the-box, and balances between requiring the management of large data set storage and configuration scalability/volatility issues (e.g., changing configuration due to data protection functionality, or other applications causing dynamic file/volume/object reconfiguration).

2. Using a combination of the reference mechanism specified in #1, but add instances to reduce / focus the scope using the addition of the `Name` property within the `Wildcards` section. This will filter the application of the scope to only the objects specifically named for each specific Wildcard reference in the URI.

This mechanism is preferred for implementations, particularly large implementations, as it provides the clients a balance of the out-of-the-box configuration provided above, with functionality available to all objects, but with the ability to apply more specifically to their known configuration. It balances the ability to automatically applies all metrics to all instances with the ability to supports scalability and volatile configurations (e.g., use of dynamic provisioning).

3. Do not use the `Wildcard` reference section, and explicitly reference the instances desired in the `Metric` section.

This model is the most inflexible, and from a client perspective, does not work well for large, dynamic configurations. Explicit references require a significant amount of manual management, and will have issues with scalability and volatility of the configuration. This approach is only suitable for small, static configurations.

While implementations may choose to support an explicit reference model, it would not be compatible with the Swordfish templates, which follow the model specified in #1 above.

3.2 Value-add Customization

The Swordfish templates, provided as part of the Swordfish bundle, follow the model for resource identification specified above to identify every potential instance of the defined property throughout their implementation, and, as the default, support the metric report to be run as-is against these.

Implementations may want to support additional client customization on top of this base functionality. This may include the type of customization described above in the “Mechanisms for Resource Identification” section to support more targeted application of the metric reports, filtering the application to only those objects of interest.

Opportunities for value-add customization may also involve allowing clients to customize in other ways, such as supporting more frequent reporting, additional notification mechanisms (e.g., `RedfishEvents`), or adding additional customized metrics, report definitions, and reports to the standard set, as well as allowing clients to create their own. All of these may be supported without relying on OEM extensions.

For More Information

More information about Swordfish and the SNIA Scalable Storage Mananagment API can be found at <http://www.snia.org/swordfish>.