Document Identifier: DSP2051

Date: 2020-03-27

Version: 1.0.0

Redfish Telemetry White Paper

Document Class: Informative

Document Status: Published

Document Language: en-US

CONTENTS

Foreword

The Redfish Telemetry White Paper was prepared by the Redfish Forum of the DMTF.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

Acknowledgments

The DMTF acknowledges the following individuals for their contributions to this document:

- George Ericson - Dell Inc.
- John Leung - Intel Corporation
- Michael Raineri - Dell Inc.

1. Introduction

Clients gather metrics remotely as the first part of the management cycle: sense, analyze, decide, and control. Telemetry are metrics that are obtained from a remote system.

Redfish is a standard designed to deliver simple and secure management for converged, hybrid IT and the Software Defined Data Center (SDDC). Both human readable and machine capable, Redfish leverages common Internet and web services standards to expose information directly to the modern tool chain.

The Redfish standard is composed of an interface protocol and a data model. The data model contains resources that express manageability capabilities and services. The Redfish telemetry model defines resources that a Redfish client can use to understand and obtain telemetry from a Redfish service.

This document describes the Redfish telemetry model for developers implementing Redfish service support or Redfish client use cases.

1.1. Use cases

The telemetry model is designed to fulfill the following use cases. The parenthetical contains the commonly known name for that class of use cases.

- Obtain the characteristics and details of a metric (metadata).
- Specify metric reports that periodically report a set of metrics (aggregation).
- Specify trigger thresholds against a metric that is monitored (monitoring).

The latter two use cases improve the efficiency of the client/service interaction. In the aggregation use case, only metrics of interest are sent to the client. In the monitoring use case, the metrics are handled locally by the service instead of being sent to the client.

1.2. Metrics in the Redfish data model

A metric is the value of a measurable quantity or quality. A metric can be an environmental sensor reading or the value of a digital counter, such as "number of bytes transferred".

In Redfish, metrics are scattered throughout the Redfish resource as **metric properties**.

The metric properties in the Redfish data model appear in a variety of ways. Figure 1 shows where metric properties exist in the Redfish data model as orange rectangles. Metric properties can be:

- JSON properties within a resource.
- JSON properties within a complex JSON object within a resource.

- Within a resource containing only metrics, such as the `MemoryMetrics` resource.

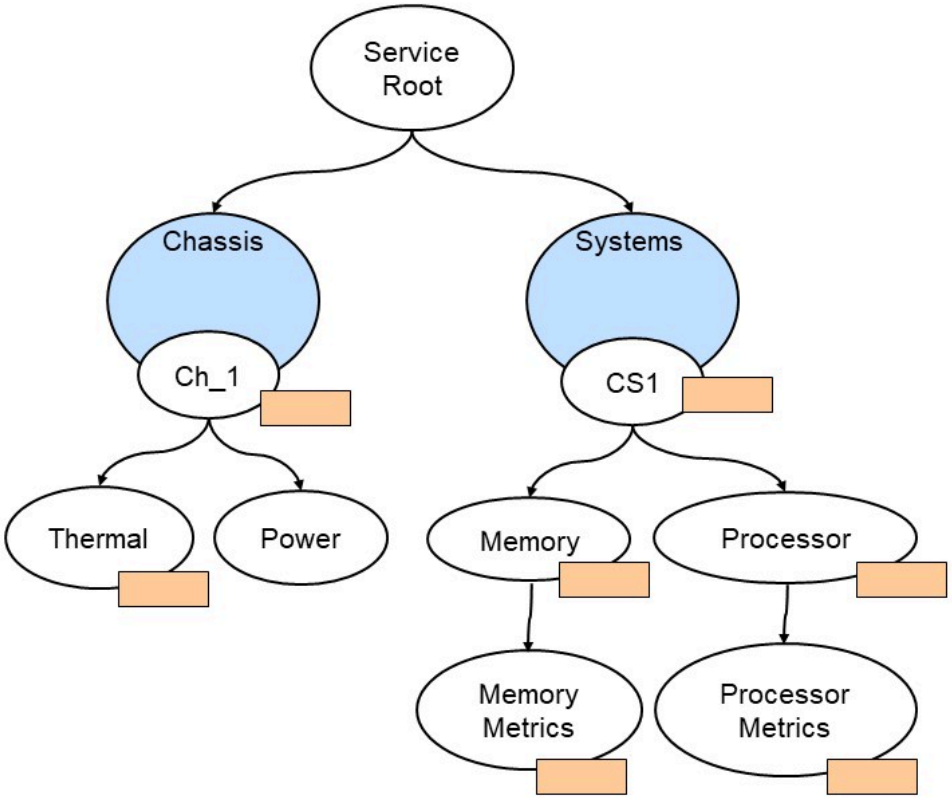

Figure 1

As Redfish is extended into new manageability domains, more metrics are added to the data model. This variation is likely to continue as resources in the data model are extended or added.

The telemetry model is designed to fulfill the above use cases, with minimal changes to current Redfish resources. The telemetry service supports **metric properties** wherever they are placed in the Redfish data model.

1.3. Types of metrics

A metric can be described by the behaviors of its values and how it is implemented.

The behavior of a metric value can be characterized. The metric definition can specify the following types of metric behaviors:

- **Numeric:** A metric whose value is any real number.
- **Discrete:** A metric whose value may have only specific discrete values. An example of a discrete metric is an inclination sensor that measures the orientation of a physical entity and can only have the values `Vertical`, `Horizontal`, and `Inclined`.
- **Gauge:** A metric whose value is continuous within a range. Most numeric and digital meter metrics are gauge metrics.
- **Counter:** A metric whose value is a non-negative integer that increases monotonically. When a counter reaches its maximum value, the value resets to 0 and resumes counting.
- **Countdown:** A metric whose value is a non-negative integer that decreases monotonically. When a countdown reaches its minimum value, the value resets to a preset value and resumes counting down.

A metric may be implemented in various ways. The metric definition can specify the following types of implementations:

- **Calculated:** A metric that is implemented by applying a calculation on one or more other metrics, where the calculation is specified. Generally, statistical metrics are calculated metrics.
- **Synthesized:** A metric that is implemented by applying a calculation on one or more other metrics, where the calculation is not specified.
- **Physical sensor:** A metric that is implemented as a physical sensor.

The above terminology can be used to more exact definition of some common metrics.

- **Environmental:** A metric that measures a characteristic of the physical world, such as temperature. The environmental metric is implemented as a physical sensor, calculated, or synthesized.
    - With a physical sensor, understanding the error between the real physical value and the measured value may be important. For a physical sensor, the metric definition can specify the precision, accuracy, and error characteristics of the physical sensor.
    - With the calculation metric, the error characteristics can be computed for the calculated

metric, from the error characteristics of the input metrics. This is known as error propagation.

- **Digital meter:** A numeric metric that is obtain my reading a digital meter. An example of a digital meter is a reading from a cache hit count register.
- **Statistical:** A calculated metric that is computed of a numeric metric over a time interval. An example of a statistical metric is the maximum power consumption.

2. Modeling of telemetry

The telemetry model is designed to fulfill two goals:

- Model is applicable to existing and future metric modeling.
- Capabilities are mutually independent.

The telemetry model is designed to be applicable to metric properties in the current Redfish data model and metric properties as the Redfish data model is extended or refactored. To this end, the telemetry model makes no requirements on the Redfish resources and their metric properties.

The telemetry model is designed to be a set of optional capabilities. The capabilities are mutually independent. A Redfish service can implement one, two, or all of the capabilities of the telemetry model. For example, metric definitions can be provided without triggers. For each metric property, an implementation can decide whether its metric definition is provided.

2.1. Telemetry service

If a Redfish service supports telemetry, the service root will contain the `TelemetryService` property.

The `TelemetryService` resource is the top level resource visible on the service root. Figure 2 shows the resources subordinate to the `TelemetryService` resource: a collection of `MetricDefinition` resources, a collection of `MetricReportDefinition` resources, a collection of `MetricReport` resources, and a collection of `Triggers` resources.

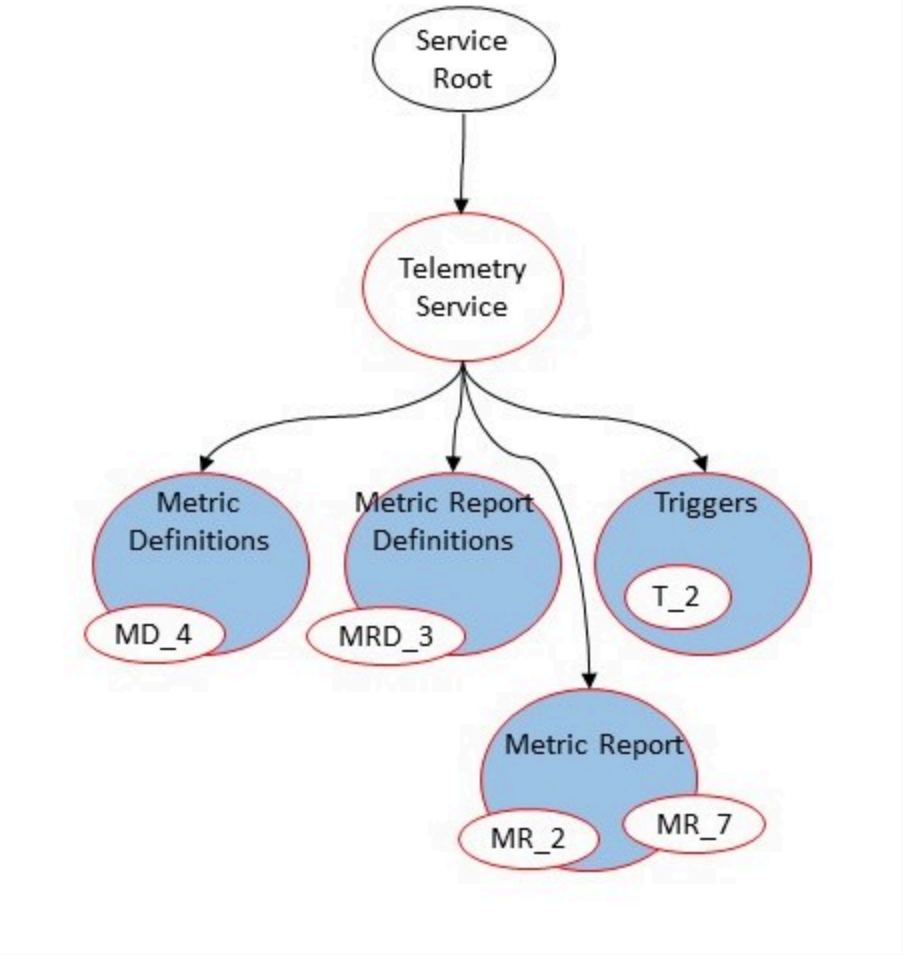

Figure 2

The `MetricDefinition` resource contains characteristics, or metadata, for metric properties. The `MetricDefinition` resource is described in the Metric definitions section.

The `MetricReportDefinition` resource contains a descriptor of the metric report to be generated. The `MetricReport` resource is the location for the report generated from a metric report definition. The `MetricReportDefinition` and `MetricReport` resources are described in the Metric report definitions section.

The `Triggers` resource contains triggers that apply to a metric property. The `Triggers` resource is described in the Triggers section.

Example `TelemetryService` resource:

```
{
    "@odata.id": "/redfish/v1/TelemetryService",
    "@odata.type": "#TelemetryService.v1_2_0.TelemetryService",
    "Id": "TelemetryService",
    "Name": "Telemetry Service",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "ServiceEnabled": true,
    "SupportedCollectionFunctions": [
        "Average",
        "Minimum",
        "Maximum"
    ],
    "MetricDefinitions": {
        "@odata.id": "/redfish/v1/TelemetryService/MetricDefinitions"
    },
    "MetricReportDefinitions": {
        "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions"
    },
    "MetricReports": {
        "@odata.id": "/redfish/v1/TelemetryService/MetricReports"
    },
    "Triggers": {
        "@odata.id": "/redfish/v1/TelemetryService/Triggers"
    }
}
```


2.2. Metric definitions

Metric definitions, defined by the `MetricDefinition` schema, contain the definition, metadata, or characteristics for a metric. A metric definition contains links to the metric properties to which the definiton applies. Figure 3 shows this reference.

In the Redfish data model, the `Power` resource contains a `PowerConsumedWatts` property, which represents "the actual power being consumed by the chassis". The metric definition that describes the `PowerConsumedWatts` metric property is a member of the metric definitions collection. The metric definition has been named `PowerconsumedWatts` to make it easier to correlate the metric property with its metric definition.

The metric definition contains references to each metric property within the Redfish data model for which the metric definition applies. This allows the same metric definition to be used to describe multiple metric properties. For example, the `PowerConsumedWatts` metric definition can reference each

`PowerConsumedWatts` property in the `Power` resource for every member in the chassis collection.

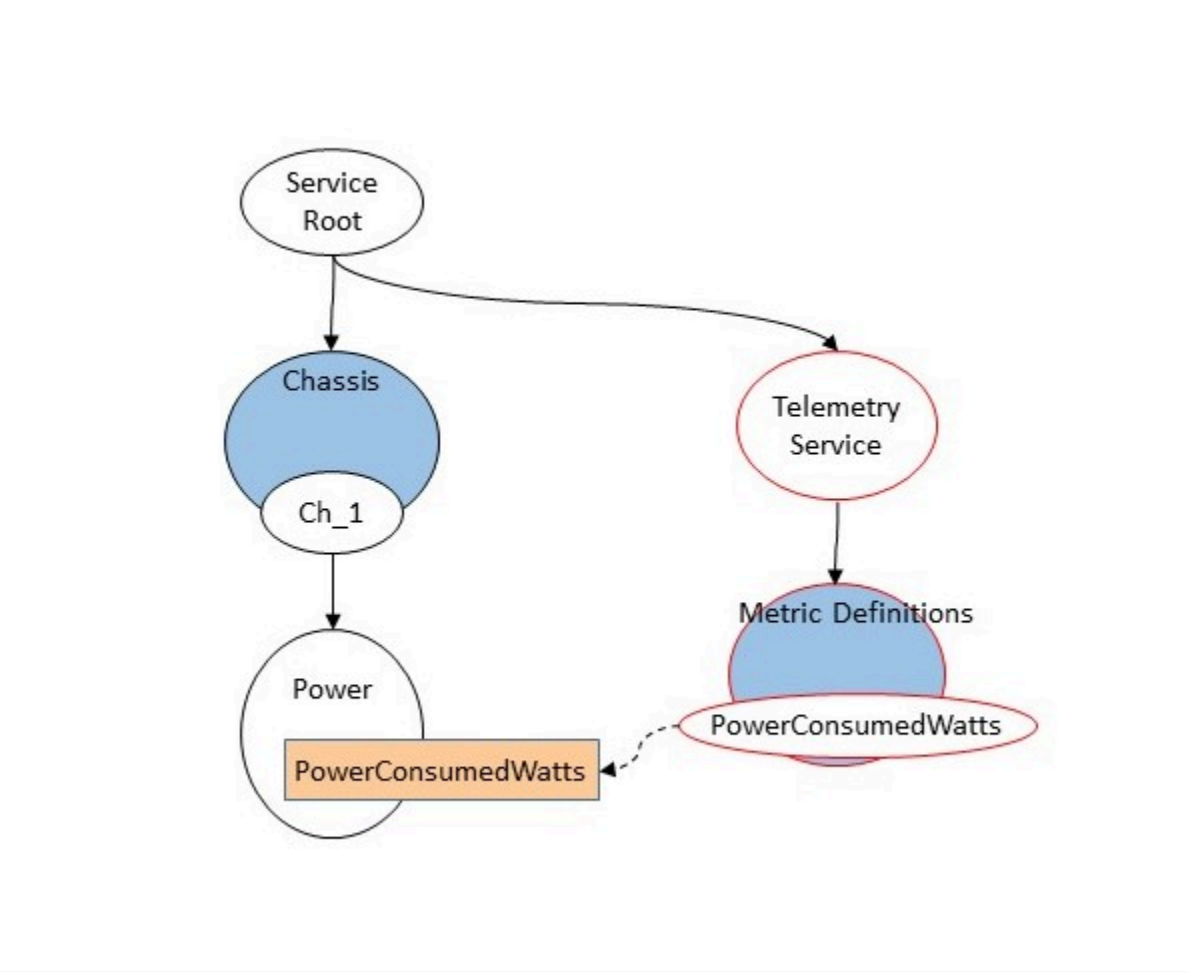

Figure 3

Within a metric definition, the properties can be grouped into four categories: context, usage, measurement, and metric references.

2.2.1. Context properties

The following properties provide context information for a Redfish client to understand the metric.

Property	Description	Values
`MetricType`	The type of metric.	• `Counter`: The metric is a monotonic counter.

Property	Description	Values
		• `Gauge`: The metric has values the can increase or decrease. • `Numeric`: The metric is a counter metric. • `Discrete`: The metric is a discrete metric.
`Implementation`	How the metric is obtained.	• `PhysicalSensor`: A physical sensor measures values of the physical environment. • `DigitalMeter`: A digital meter measures values of digital elements, such as cache hits. • `Calculated`: A calculated metric is obtained by applying a calculation on other metrics. • `Synthesized`: A synthesized metric is a calculated metric that represents physical sensor value.
`PhysicalContext`	The physical context of the metric.	See the `PhysicalContext` schema.

2.2.2. Usage properties

The following properties provide guidance on how the metric can be used by a Redfish client.

Property	Description	Values
`Calculable`	Types of calculations that can be applied to the metric reading.	• `NonCalculatable`: No calculations should be performed on the metric reading. • `Summable`: The sum of the metric reading across multiple instances is meaningful. • `NonSummable`: The

Property	Description	Values
		sum of the metric reading across multiple instances is not meaningful.
`Calibration`	Calibration offset that clients add to the metric reading to obtain the actual value.	A decimal value.
`IsLinear`	Whether the metric readings are linear, as opposed to non-linear. Linear metrics may be compared using a lesser-than or greater-than relationship.	A boolean.
`MaxReadingRange`	Highest possible value for the metric readings.	A decimal value.
`MetricDataType`	Data type of the metric reading.	• `Boolean`: JSON boolean. • `String`: JSON string. • `Decimal`: JSON decimal. • `Integer`: JSON integer. • `String`: JSON string. • `Enumeration`: JSON string with a set of enumerations defined.
`MinReadingRange`	Lowest possible value for the metric readings.	A decimal value.

2.2.3. Measurement properties

These properties characterize the measurement itself.

The following measurement properties were obtained from the Energy Efficient HPC WG's PowerAPI

specification.

Property	Description
`Accuracy`	Estimated percent error +/- of measured vs. actual values.
`Precision`	Number of significant digits in values.
`TimestampAccuracy`	Accuracy of returned timestamps, represented as an ISO8601 duration.

The following measurement properties are other properties defined by the Redfish telemetry model.

Property	Description
`CalculationAlgorithm`	Calculation performed to obtain the metric.
`CalculationTimeInterval`	Time interval over which the calculation is performed on the metric readings.
`DiscreteValues`	Possible values of a discrete metric.
`SensingInterval`	Time interval between when a metric is updated.
`Units`	Units of the metric, as defined by Unified Code for Units of Measure (UCUM).

2.2.4. Metric references properties

The `MetricDefinition` resource contains a list that references the metric properties to which the metric definition applies. A metric definition may apply to many metric properties throughout the Redfish data model. This would result in the `MetricProperties` property containing a very large list of references to each metric property.

In order to reduce the size of this list, an optional wildcard syntax may be used. In the syntax, the `Wildcards` property is used in conjunction with the `MetricProperties` property.

This wildcard mechanism is also used in the `MetricReportDefinition` and `Triggers` resources.

2.2.4.1. MetricProperties

The `MetricProperties` property contains a list of each metric property described by the metric definition. The `MetricProperties` entry may contain wildcards in the string. The wildcard is expressed by curly braces `{}`.

```
{
    "MetricProperties": [
        "/redfish/v1/Chassis/{ChassisID}/Power#/PowerControl/0/PowerConsumedWatts",
        "/redfish/v1/Chassis/{ChassisID}/Power#/PowerControl/1/PowerConsumedWatts"
    ]
}
```

2.2.4.2. Wildcards

The `Wildcards` property contains a list of the replacement list for wildcards. Each wildcard entry contains the wildcard used in the `MetricProperties` entry and the value with which to replace it. The value `*` means all possible values.

The following `Wildcards` property specifies that the string `{ChassisID}` in each `MetricProperties` entry is to be replaced by the values `1`, `2`, and `3`.

```
{
    "Wildcards": [
        {
            "Name": "ChassisID",
            "Values": [
                "1",
                "2",
                "3"
            ]
        }
    ]
}
```

2.2.5. Metric definition examples

2.2.5.1. PowerConsumedWatts within the Power resource

The following example contains a metric definition for the `PowerConsumedWatts` property found in the `Power` resource.

The `Power` resource contains a `PowerControl` array property. Each entry in the array contains a `PowerConsumedWatts` property. The `MetricProperties` property in the metric definition specifies two `PowerConsumedWatts` properties from the `Power` resource.

```
{
```

```
    "@odata.id": "/redfish/v1/TelemetryService/MetricDefinitions/PowerConsumedWatts",
    "@odata.type": "#MetricDefinition.v1_0_3.MetricDefinition",
    "Id": "PowerConsumedWatts",
    "Name": "Power Consumed Watts Metric Definition",
    "MetricType": "Numeric",
    "Implementation": "PhysicalSensor",
    "PhysicalContext": "PowerSupply",
    "MetricDataType": "Decimal",
    "Units": "W",
    "Precision": 4,
    "Accuracy": 1,
    "Calibration": 2,
    "MinReadingRange": 0,
    "MaxReadingRange": 50,
    "SensingInterval": "PT1S",
    "TimestampAccuracy": "PT1S",
    "Wildcards": [
        {
            "Name": "ChassisID",
            "Values": [
                "1",
                "2",
                "3"
            ]
        }
    ],
    "MetricProperties": [
        "/redfish/v1/Chassis/{ChassisID}/Power#/PowerControl/0/PowerConsumedWatts",
        "/redfish/v1/Chassis/{ChassisID}/Power#/PowerControl/1/PowerConsumedWatts"
    ]
}
```


2.3. Metric report definitions

The Redfish service may support the ability to generate metric reports. Metric report definitions, defined by the `MetricReportDefinition` schema, specify the metric reports that are generated.

A metric report is created by the Redfish service and can be transmitted using the Redfish event service, stored locally for retrieval, or both. A metric report can be created periodically, when a reading value changes, or upon request. A common usage for a metric report is to aggregate metric readings.

The `MetricReportDefinition` resource specifies the contents and periodicity of the metric report. It also contains links to the metric properties to which the definiton applies. Figure 4 shows this reference.

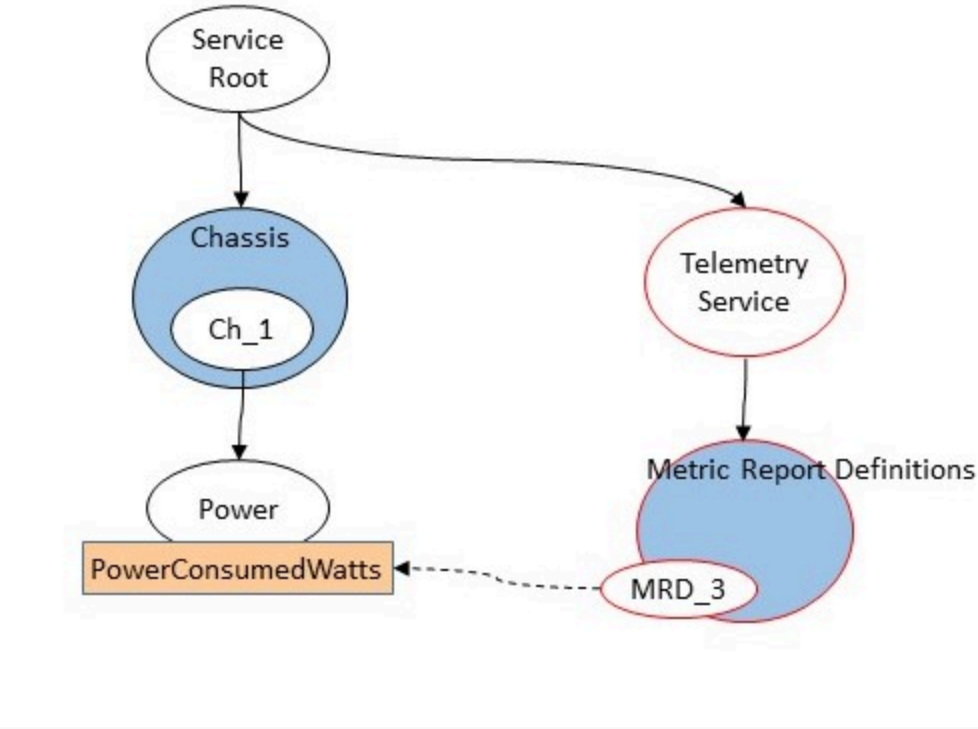

Figure 4

Within a metric report definition, the properties can be grouped into four categories: context, when to generate, disposition, and metric references.

2.3.1. Context properties

The following properties provide context information for a Redfish client to understand the metric report.

Property	Description	Values
`MetricReportDefinitionType`	When the report is created.	• `Periodic`: The metric report is generated periodically. • `OnChange`:

Property	Description	Values
		The metric report is generated when the metric values change. • `OnRequest`: The metric report is generated when a HTTP GET is performed on on the specified metric report.
`MetricReportDefinitionEnabled`	Whether the metric report definition is enabled. When enabled, metric reports will be generated based other properties within the metric report definition.	A boolean.

2.3.2. When to generate

If the `MetricReportDefinitionType` property contains the value `Periodic`, the `Schedule` property specifies when to generate the metric report. The `RecurrenceInterval` property within `Schedule` determines how frequently the metric report is generated. Other properties can exist in `Schedule` to control other aspects of when the metric report is generated.

The following example specifies the metric report is generated every 60 seconds, but stops after 100 occurrences.

```
{
    "MetricReportDefinitionType": "Periodic",
    "Schedule": {
        "RecurrenceInterval": "PT60S",
```

```
        "MaxOccurrences": 100
    },
    ...
}
```

2.3.3. Disposition

The following properties define what a service does with the metric values.

Property	Description	Values
`ReportActions`	Specifies the actions to perform when a metric report is generated.	• `LogToMetricReportsCollection`: When a metric report is scheduled to be generated, record the occurrence to the metric report collection. • `RedfishEvent`: When a metric report is scheduled to be generated, send a Redfish event message of type `MetricReport`.
`ReportUpdates`	Specifies what to do with subsequent metric reports when a metric report already exists.	• `Overwrite`: When a metric report is updated, overwrite the metric report. • `AppendWrapsWhenFull`: When a metric report is updated, new information is appended to the report. The metric report overwrites its entries with new entries when the metric report has reached its maximum capacity. • `AppendStopsWhenFull`: When a metric report is updated, append to the specified metric report. This also indicates that the metric report stops adding entries when the metric report has reached its maximum capacity. • `NewReport`: When a metric report is updated, create a new metric report, whose resource name is the metric

Property	Description	Values
		report resource name concatenated with the timestamp.
`CalculationAlgorithm`	Specifies the function to apply to the list of metric properties.	• `Average`: The metric is calculated as the average of the metric readings over a duration. • `Maximum`: The metric is calculated as the maximum of the metric readings over a duration. • `Minimum`: The metric is calculated as the minimum of the metric readings over a duration. • `Summation`: The metric is calculated as the sum of the metric readings over a duration.
`CollectionTimeScope`	Specifies the scope of the metric values.	• `Point`: The metric values apply to a point in time, specified by the `Timestamp` property. • `Interval`: The metric values apply to an interval of time, specified by the `Timestamp` and `CollectionDuration` properties. • `StartupInterval`: The metric values to a time interval that began at the startup of the measured resource.

2.3.4. Metric references properties

The `MetricReportDefinition` resource contains a list that references metric properties within the Redfish data model to include or the resource can specify the calculation to be performed on a metric property. Therefore, the `MetricProperties` property may appear as a property of the resource or as a property within the `Metrics` property. For both these usages, the wildcard mechanism described in the Metric definition section can be used.

2.3.4.1. MetricProperties

The `MetricProperties` array property specifies metrics that are included in the metric report. The `MetricProperties` property may have wildcards.

The following JSON fragment specifies the `AverageConsumedWatts` property in item `0` of the `PowerControl` property for two chassis instances, `Tray_1` and `Tray_2`.

```
{
    "Wildcards": [
        {
            "Name": "PWild",
            "Values": [
                "0"
            ]
        },
        {
            "Name": "TWild",
            "Values": [
                "Tray_1",
                "Tray_2"
            ]
        }
    ],
    "MetricProperties": [
        "/redfish/v1/Chassis/{TWild}/Power#/PowerControl/{PWild}/PowerConsumedWatts/
PowerMetrics/AverageConsumedWatts"
    ]
}
```

2.3.4.2. Metrics

While the above can be used when the Redfish model has a property for the metric of interest, such as `AverageConsumedWatts`, there a many situations where a property does not exist. For those situations, the `Metrics` property can be used to specify the calculation to be performed on a metric property that does exist in the Redfish data model.

The `Metrics` property specifies metrics included in the metric report that are derived by applying the algorithm specified calculation to each of the metric properties listed. The `Metrics` property is an array of objects that contain the following properties.

Property	Description
`MetricId`	Label for the metric.

Property	Description
`MetricProperties`	List of URIs for the metric properties on which the calculation is made.
`CollectionFunction`	Function to perform on each of the metric properties listed.
`CollectionDuration`	Duration over which the function is computed.
`CollectionTimeScope`	Scope of time over which the function is applied.

The following JSON fragment specifies the average be calculated over a one second interval using the `PowerConsumedWatts` property in item 0 of the `PowerControl` property.

```
{
    "Wildcards": [
        {
            "Name": "PWild",
            "Values": [
                "0"
            ]
        },
        {
            "Name": "TWild",
            "Values": [
                "Tray_1",
                "Tray_2"
            ]
        }
    ],
    "Metrics": [
        {
            "MetricId": "AverageConsumeWatts",
            "CollectionFunction": "Average",
            "CollectionTimeScope": "Interval",
            "CollectionDuration": "PT1S",
            "MetricProperties": [
                "/redfish/v1/
Chassis/{TWild}/Power#/PowerControl/{PWild}/PowerConsumedWatts"
            ]
        }
    ]
}
```

2.3.5. Metric report definition examples

2.3.5.1. Power statistics

This metric report definition specifies a metric report that includes the power statistics, using the `AvgPowerConsumedWatts`, `MinPowerConsumedWatts`, and `MaxPowerConsumedWatts` properties, for the system power and CPU power domains on three platforms, `Tray_1`, `Tray_2`, and `Tray_3`.

The metric report is specified to be generated every one tenth of a second. It is also specified to be transmitted as an event, and also placed in the metric reports collection. When logging, the metric report should overwrite any previous metric report.

```
{
    "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/PowerMetrics",
    "@odata.type": "#MetricReportDefinition.v1_3_0.MetricReportDefinition",
    "Id": "PowerMetrics",
    "Name": "Transmit and Log Power Metrics",
    "MetricReportDefinitionType": "Periodic",
    "MetricReportDefinitionEnabled": true,
    "Schedule": {
        "RecurrenceInterval": "PT0.1S"
    },
    "ReportActions": [
        "RedfishEvent",
        "LogToMetricReportsCollection"
    ],
    "ReportUpdates": "Overwrite",
    "MetricReport": {
        "@odata.id": "/redfish/v1/TelemetryService/MetricReports/PowerMetrics"
    },
    "Status": {
        "State": "Enabled"
    },
    "Wildcards": [
        {
            "Name": "PWild",
            "Values": [
                "0",
                "1"
            ]
        },
        {
            "Name": "TWild",
            "Values": [
                "Tray_1",
                "Tray_2",
```

```
                "Tray_3"
            ]
        }
    ],
    "MetricProperties": [
        "/redfish/v1/Chassis/{TWild}/Power#/PowerControl/{PWild}/PowerMetrics/
AverageConsumedWatts",
        "/redfish/v1/Chassis/{TWild}/Power#/PowerControl/{PWild}/PowerMetrics/
MinConsumedWatts",
        "/redfish/v1/Chassis/{TWild}/Power#/PowerControl/{PWild}/PowerMetrics/
MaxConsumedWatts"
    ]
}
```

2.3.5.2. Platform power usage

This metric report definition specifies a metric report that includes the power consumption, using the `PowerConsumedWatts` property, for the system on two platforms, `Tray_1` and `Tray_2`.

The metric report is specified to be generated every one hour. It is also specified to be transmitted as an event, and also placed in the metric reports collection. When logging, the metric report contains at most 256 entries, and older entries are discarded when the metric report is full.

```
{
    "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/
PlatformPowerUsage",
    "@odata.type": "#MetricReportDefinition.v1_3_0.MetricReportDefinition",
    "Id": "PlatformPowerUsage",
    "Name": "Transmit and Log Platform Power Usage",
    "MetricReportDefinitionType": "Periodic",
    "MetricReportDefinitionEnabled": true,
    "Schedule": {
        "RecurrenceInterval": "PT1H"
    },
    "ReportActions": [
        "RedfishEvent",
        "LogToMetricReportsCollection"
    ],
    "ReportUpdates": "AppendWrapsWhenFull",
    "AppendLimit": 256,
    "MetricReport": {
        "@odata.id": "/redfish/v1/TelemetryService/MetricReports/PlatformPowerUsage"
    },
    "Status": {
        "State": "Enabled"
```

```
    },
    "Wildcards": [
        {
            "Name": "PWild",
            "Values": [
                "0"
            ]
        },
        {
            "Name": "TWild",
            "Values": [
                "Tray_1",
                "Tray_2"
            ]
        }
    ],
    "MetricProperties": [
        "/redfish/v1/Chassis/{TWild}/Power#/PowerControl/{PWild}/PowerConsumedWatts"
    ]
}
```

2.4. Metric reports

Metric reports, defined by the `MetricReport` schema, contain the metric readings and any metadata associated with the readings. Depending on the configuration of the appropriate metric report definition, metric reports can be found within the metric report resource collection of the telemetry service, or they can be sent asynchronously as a Redfish event.

The following properties can be found in a metric report.

Property	Description
`MetricReportDefinition`	A link to the metric report definition that generated this metric report.
`MetricValues`	An array of objects containing metric readings that were captured.
`Timestamp`	The date and time when the metric report was created. This is not necessarily when the metric readings themselves were sampled.

The `MetricValues` array contains a list of the readings that have been captured by the service that pertain to the metric report definition. Depending on the configuration of the metric report definition, the metric report can contain historic data. The following properties can be found within each index of the `MetricValues` array.

Property	Description
`MetricId`	Label for the metric.
`MetricValue`	The string representation of the metric reading. It should be noted that even booleans and numbers are encoded as strings, which means clients will need to decode the value based on the metric definition.
`Timestamp`	The date and time when the metric was sampled by the service.
`MetricProperty`	The URI for the property from which this metric is derived.
`MetricDefinition`	A link to the metric definition that provides the characteristics of the metric property.

2.4.1. Metric report examples

2.4.1.1. Platform power usage

The following metric report contains samples of the `PlatformConsumedWatts` property for two chassis instances, `Tray_1` and `Tray_2`. There are a total of six available readings, with three per chassis instance. From the timestamps, each pair of readings was taken one hour apart.

```
{
    "@odata.id": "/redfish/v1/TelemetryService/MetricReports/PlatformPowerUsage",
    "@odata.type": "#MetricReport.v1_3_0.MetricReport",
    "Id": "PlatformPowerUsage",
    "Name": "PlatformPowerUsage",
    "MetricReportDefinition": {
        "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/
PlatformPowerUsage"
    },
    "MetricValues": [
        {
            "Timestamp": "2016-11-08T12:25:00-05:00",
            "MetricValue": "103",
            "MetricProperty": "/redfish/v1/Chassis/Tray_1/Power#/PowerControl/0/
PlatformConsumedWatts"
        },
        {
            "Timestamp": "2016-11-08T12:25:00-05:00",
            "MetricValue": "103",
            "MetricProperty": "/redfish/v1/Chassis/Tray_2/Power#/PowerControl/0/
PlatformConsumedWatts"
        },
```

```
        {
            "Timestamp": "2016-11-08T13:25:00-05:00",
            "MetricValue": "106",
            "MetricProperty": "/redfish/v1/Chassis/Tray_1/Power#/PowerControl/0/
PlatformConsumedWatts"
        },
        {
            "Timestamp": "2016-11-08T13:25:00-05:00",
            "MetricValue": "106",
            "MetricProperty": "/redfish/v1/Chassis/Tray_2/Power#/PowerControl/0/
PlatformConsumedWatts"
        },
        {
            "Timestamp": "2016-11-08T14:25:00-05:00",
            "MetricValue": "107",
            "MetricProperty": "/redfish/v1/Chassis/Tray_1/Power#/PowerControl/0/
PlatformConsumedWatts"
        },
        {
            "Timestamp": "2016-11-08T14:25:00-05:00",
            "MetricValue": "107",
            "MetricProperty": "/redfish/v1/Chassis/Tray_2/Power#/PowerControl/0/
PlatformConsumedWatts"
        }
    ]
}
```


2.5. Triggers

The Redfish service can support the ability to specify a set of triggers or thresholds for a list of metric properties. The `Triggers` resource specifies the trigger thresholds that apply to the listed metrics. A trigger can result in one or more actions, such as an alert being transmitted using the event service or an event logged in the log service.

The `Triggers` resources allows for representing a common set of triggers to be applied to multiple metric properties. This removes the need for a client to apply thresholds in each of the resources where the metric property resides.

Figure 5 shows the `Triggers` resource for to the `PowerConsumedWatts` property in the `Power` resource.

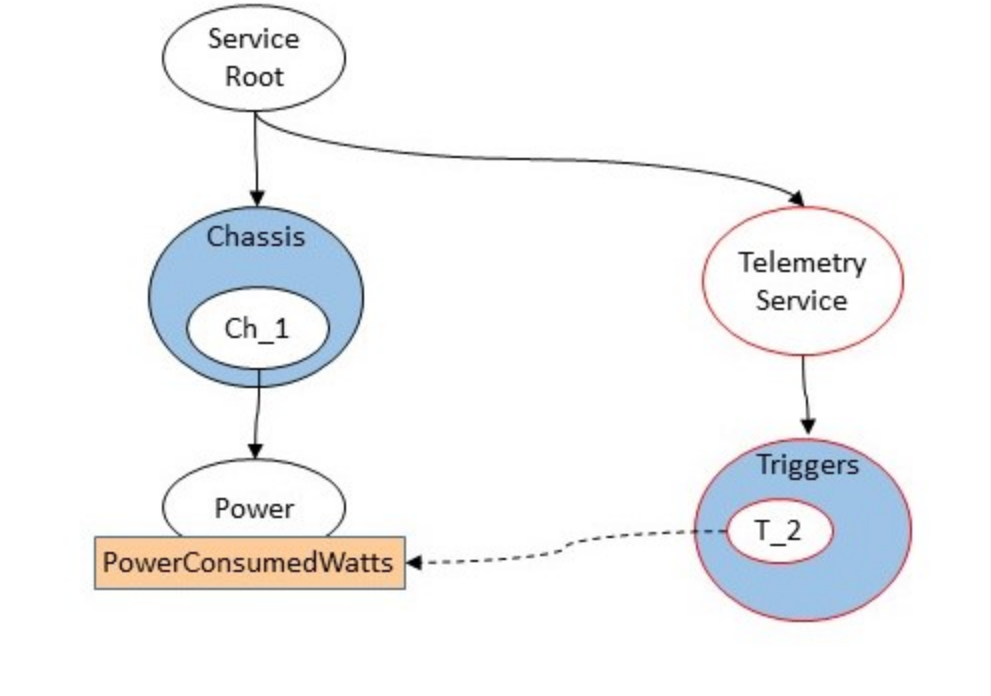

Figure 5

2.5.1. Common properties

The following properties are common to all `Triggers` resources.

Property	Description	Values
`MetricType`	Type of metrics being monitored and indicates other properties that should be present.	• `Numeric`: The triggers are numeric. See Numeric triggers. • `Discrete`: The triggers are discrete. See Discrete triggers.
`TriggerActions`	Actions to perform when a trigger condition is met.	• `LogToLogService`: Log the trigger condition to the log service. • `RedfishEvent`: Send the trigger as a Redfish event to subscribers. • `RedfishMetricReport`: Force an

Property	Description	Values
		update of the specified metric reports.
`MetricProperties`	Metrics that are monitored against the triggers.	An array of strings containing URIs to properties in the Redfish data model. Wildcards may also be specified. See Metric references properties for details.

2.5.2. Numeric triggers

If the `MetricType` property contains the value `Numeric`, the `NumericThresholds` property is used to specify the thresholds for a numeric trigger to occur. The `NumericThresholds` property contains the following thresholds.

Property	Description
`UpperWarning`	Value at which the reading is above normal range.
`UpperCritical`	Value at which the reading is above normal range and requires attention.
`LowerWarning`	Value at which the reading is below normal range.
`LowerCritical`	Value at which the reading is below normal range and requires attention.

Each of the above threshold properties are objects that contain the following properties.

Property	Description
`Reading`	Value of the threshold.
`Activation`	Direction that the threshold is crossed to trip the trigger.
`DwellTime`	Duration that value must be over the the threshold before the trigger is tripped.

The following JSON fragment specifies two numeric triggers: an upper critical trigger with the value `50` and an upper warning trigger with the value `48.1`. Both are specified to trigger when the value is increasing above the threshold.

```
{
```

```
    "NumericThresholds": {
        "UpperCritical": {
            "Reading": 50,
            "Activation": "Increasing",
            "DwellTime": "PT0.001S"
        },
        "UpperWarning": {
            "Reading": 48.1,
            "Activation": "Increasing",
            "DwellTime": "PT0.004S"
        }
    }
}
```

2.5.3. Discrete triggers

If the `MetricType` property contains the value `Discrete`, the `DiscreteTriggerConditions` and `DiscreteTriggers` properties are used to specify one or more discrete triggers.

The `DiscreteTriggerConditions` property specifies the conditions for a trigger to occur, and can contain the following values:

- `Specified`: A trigger occurs when the value of the metric becomes one of the values listed in the `DiscreteTriggers` property.
- `Changes`: A trigger occurs whenever the value of the metric changes.

The `DiscreteTriggers` property is an array that specifies the values of metric that will trip the trigger, if the `DiscreteTriggerCondition` property contains the value `Specified`. Each member of the `DiscreteTriggers` array is an object containing the following properties.

Property	Description
`Name`	Name of the trigger.
`Value`	Discrete metric value that constitutes a trigger event.
`DwellTime`	Duration that a trigger event persists before the trigger is tripped.
`Severity`	Value of the `Severity` property within the Redfish event payload.

The following JSON fragment specifies a discrete trigger when an orientation metric contains the value `Inclined`. The orientation metric may have the values `Vertical`, `Horizontal`, and `Inclined`.

```
{
    "DiscreteTriggerCondition": "Specified",
    "DiscreteTriggers": [
        {
            "Value": "Inclined",
            "DwellTime": "PT0.001S",
            "Severity": "Warning"
        }
    ]
}
```

2.5.4. Triggers Example

The following is an example of a numeric trigger with two thresholds: upper critical and upper warning. When the trigger occurs, a Redfish event is sent to the subscribers.

```
{
    "@odata.id": "/redfish/v1/TelemetryService/Triggers/PlatformPowerCapTriggers",
    "@odata.type": "#Triggers.v1_1_1.Triggers",
    "Id": "PlatformPowerCapTriggers",
    "Name": "Triggers for platform power consumed",
    "MetricType": "Numeric",
    "TriggerActions": [
        "RedfishEvent"
    ],
    "NumericThresholds": {
        "UpperCritical": {
            "Reading": 50,
            "Activation": "Increasing",
            "DwellTime": "PT0.001S"
        },
        "UpperWarning": {
            "Reading": 48.1,
            "Activation": "Increasing",
            "DwellTime": "PT0.004S"
        }
    },
    "MetricProperties": [
        "/redfish/v1/Chassis/1/Power#/PowerControl/0/PowerConsumedWatts"
    ]
}
```

3. Appendix

3.1. References

- PowerAPI Specification, Energy Efficient HPC Working Group, https://powerapi.sandia.gov
- The Unified Code for Units of Measure, Unified Code for Units of Measure (UCUM) Organization, http://www.unitsofmeasure.org/ucum.html
- Date and time format - ISO 8601, International Organization for Standardization, https://www.iso.org/standard/40874.html

3.2. Change log

Version	Date	Description
1.0.0	2020-03-27	Initial release.