



Document Identifier: DSP2051

Date: 2018-01-03

Version: 0.1.0

Redfish Telemetry White Paper

Information for Work-in-Progress version:

IMPORTANT: This document is not a standard. It does not necessarily reflect the views of the DMTF or its members. Because this document is a Work in Progress, this document may still change, perhaps profoundly and without notice. This document is available for public review and comment until superseded.

Provide any comments through the DMTF Feedback Portal: <http://www.dmtf.org/standards/feedback>

Document Class: Informative

Document Status: Work-in-Progress

Document Language: en-US

Copyright Notice

Copyright © 2018 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit <http://www.dmtf.org/about/policies/disclosures.php>.

This document's normative language is English. Translation into other languages is permitted.

1. Introduction.....	6
2. Requirements of Telemetry Model.....	6
3. Telemetry Model	8
3.1. Modeling Options	8
3.1.1. Specifying Re-occurring Measurements.....	9
3.1.2. Specifying Triggers	9
3.1.3. Specifying Calculations	10
3.1.4. Specifying Duration	11
3.2. Telemetry Service	12
3.3. Metric Definitions	13
3.3.1. Metric Type	15
3.3.2. ImplementationType	15
3.3.3. SensorType	16
3.3.4. PhysicalContext.....	16
3.3.5. Units.....	16

3.3.6. DiscreteValues	16
3.3.7. Precision	16
3.3.8. Accuracy	16
3.3.9. TimeStampLatency	16
3.3.10. TimeStampAccuracy	16
3.3.11. TimeWindow	17
3.3.12. UpdateRate	17
3.3.13. SampleRate	17
3.3.14. MeasureMethod	17
3.3.15. Wildcards	17
3.3.16. Example	18
3.4. Metric Report Definitions	18
3.4.1. MetricReportType	19
3.4.2. ReportActions	19
3.4.3. MetricProperties	20
3.4.4. Example	18
3.5. Triggers	21
3.5.1. TriggerType	21
3.5.2. TriggerActions	21
3.5.3. NumericTriggers	22
3.5.4. DiscreteTriggerConditions	22
3.5.5. DiscreteTriggers	22
3.5.6. MetricProperties	20
3.5.7. Example	18
4. References	23

Foreword

The Redfish Telemetry White Paper was prepared by the Scalable Platforms Management 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

1. Introduction

Redfish is an evolving hardware management standard that is designed to be flexible, extensible, and inter-operable.

Redfish supports a wide variety of solutions, from servers to storage, networking to fabrics with power and cooling. As such, each management domain has models in which readings for important data are scattered throughout the model. The Redfish telemetry model makes it possible to describe the a since locations where the metadata for the readings/metrics, the definition of reports gathering readings from multiple locations in the model, and specifying triggers and trigger actions associated with a specific metric reading.

The telemetry model is a proposal to support the ability for a Redfish service to expose:

- The characteristics of metrics
- The triggers to apply to a specific metric
- The characteristics and contents of metric reports

The mockup and schema for the telemetry model was released as work-in-progress v0.9a [Redfish Telemetry Model Proposal]. This whitepaper describes that model.

In specifying this work-in-progress telemetry model, there are some aspects of the telemetry model where two models can be used to model an aspect of telemetry. Each model would have benefits and disadvantages. The [Modeling Options](#) section describes where the modeling options which appear in this document.

Feedback on the desirability of one or both modeling options are desired. Please provide feedback, since the feedback will inform and influence the specification of an unified telemetry model.

2. Requirements of Telemetry Model

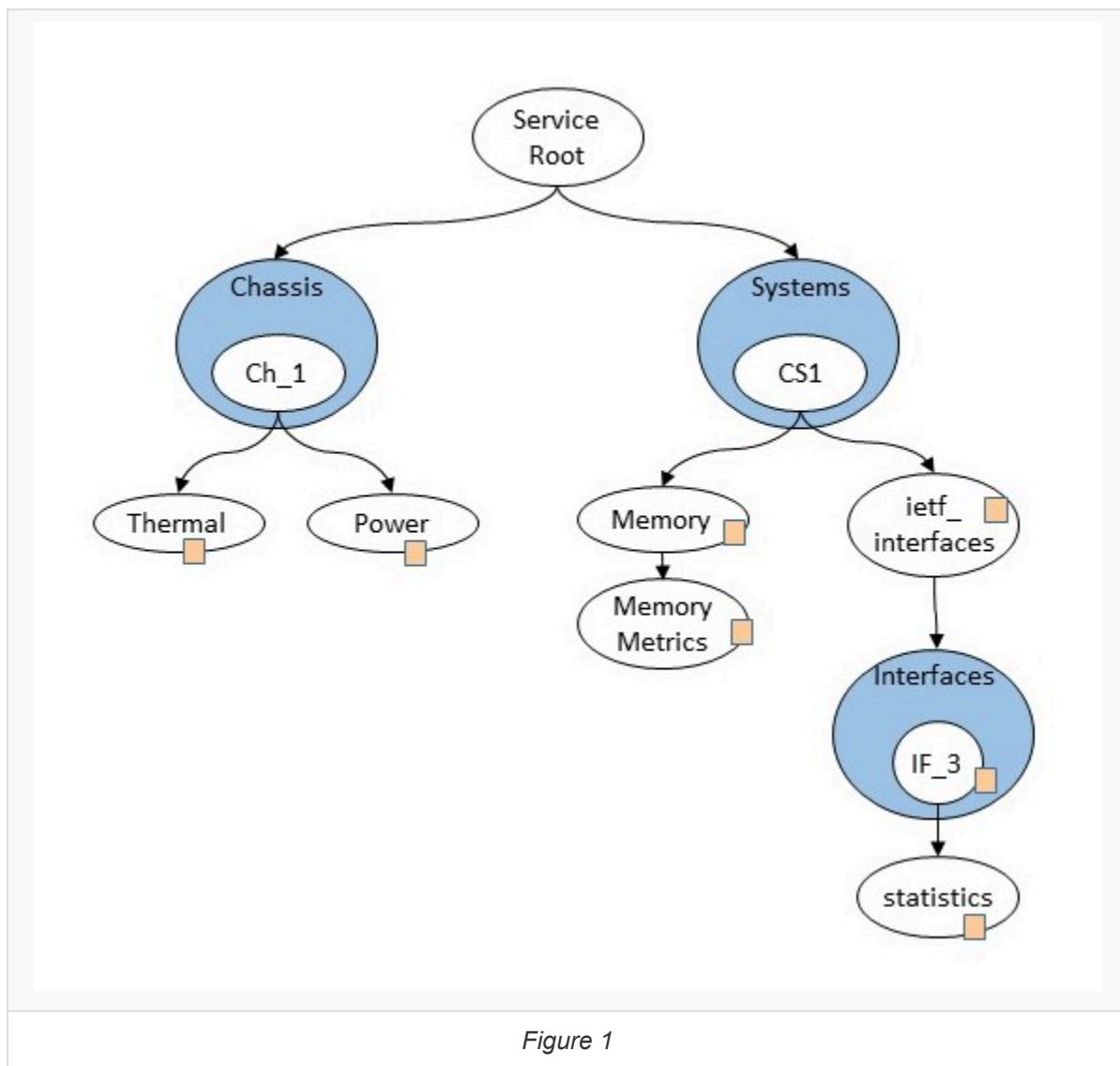
The telemetry model is designed to fulfill the following requirements. These requirements are expounded subsequently.

- The telemetry model include optional extensions to the existing model
- Support existing metric properties in the models
- Support metrics from various type of sensors
- Models for capabilities beyond a telemetry reading

The requirement that the telemetry model include optional extensions to the existing model means a Redfish service can implement one, two, or all of the capabilities of the telemetry model.

The requirement to support existing properties within the models means that properties is driven by the fact that properties representing metrics, metric metadata and sensor characteristics already exist in some resource definitions. In this document, these properties will be referred to collectively as *metric properties*.

The metric properties in the Redfish model appear in a variety of ways. Figure 1 shows where metric properties exist in the Redfish models. Metric properties can be a simple JSON object or within a complex JSON object within a resource (e.g. Thermal and Power resource). Metric properties can be in a dedicated resource (e.g. MemoryMetrics resource). With YANG models, the dedicated resource is used often (e.g. statistics resource). The presumption is that this variety will continue as models are extended or added.



The requirement that the telemetry model supports various telemetry resources recognizes that there are many sources of metrics. The telemetry model supports the following types of metrics:

- **Environmental sensor** - measures an element in the physical world (e.g. temperature). These metrics are characterized by a physical value and the measurement of that value. The sensitivity of the sensor determines the precision, accuracy and error characteristics of the measured value.
- **Digital meter** - measures a value that characterizes digital behavior (e.g. cache hits). A cache hit meter is an example of a digital meter.
- **Discrete sensor** - measures specific discrete values. An inclination sensor which measures the orientation of a physical resource (vertical, horizontal, inclined) is an example of a discrete sensor.
- **Gauge sensor** - whose measurements are continuous within a range. An environmental sensor is an example of a gauge sensor.
- **Statistical metric** - is a computation on a metric over a time interval (e.g. min, max, average)
- **Synthesized metric** - is a computation of a metric from other metrics. These metrics are calculated by applying a formula to one or more other metrics. The precision, accuracy and error of this metric is influenced by the precision, accuracy and error of its dependent metrics.

The requirement to support higher level telemetry capabilities means supporting more than just metric readings. This includes:

- Ability to get aggregated metrics
- Ability to specify thresholds against which a metric is monitored
- Ability to subscribe for alerts about threshold crossing

3. Telemetry Model

The telemetry model has a Telemetry Service with four subordinate collection resources for:

- Metric Definitions - this contains the definition of metric properties (characteristics, metadata)
- Metric Report Definitions - this contains definition of metric reports to be generated
- Metric Reports - this contains metric reports, if logging of metric reports are requested
- Metric Triggers - this contains threshold triggers and actions that apply to a metric property

3.1. Modeling Options

There are some areas of the Telemetry for which two ways of modeling an aspect of telemetry are possible and each option has benefits and disadvantages. In those cases, both models are presented in the telemetry model and described in this document.

Feedback from the industry is requested so a decision on which model to retain in the Redfish standard.

The areas where modeling options exists:

- Specifying re-occurring measurements
- Specifying triggers
- Specifying calculations
- Specifying durations

3.1.1. Specifying Re-occurring Measurements

There are two locations for specifying the re-occurring measurements. The options are in the MetricDefinition resource or the MetricReportDefinition resource.

When specified in the MetricDefinition resource, there is a single SensingInterval property.

```
"SensingInterval": "PT0.001S",
```

When specified in the MetricReportDefinition resource, the re-occurrence property is placed in the RecurrenceInterval property within the Schedule object. The Lifetime property within the Schedule object is used to control a schedule. The property specifies "the time after provisioning when the schedule as a whole expires" (see Schedule_v1.xml).

```
"Schedule": {  
  "Lifetime": "P05D",  
  "RecurrenceInterval": "PT0.001S"  
},
```

3.1.2. Specifying Triggers

There are two locations for specifying the triggers. The options are in the Triggers resource or the MetricReportDefinition resource.

When specified in the Triggers resource, the TriggerCondition is place in a member of the Triggers collection resource. The Triggers resource model is described in the [Triggers](#) section.

When specified in the MetricReportDefinition resource, the TriggerCondition property is placed within the Metrics object.

```
"Metrics": [  
  {  
    "MemberID": "PowerUsageReading",  
    "MetricProperties": [ "/redfish/v1/Chassis/...PowerConsumedWatts" ],
```

```

    "CollectionDuration": "PT0.020S",
    "TriggerCondition": {
      "DwellInterval": "PT0.001S",
      "TriggerType": "Numeric",
      "NumericTriggerConditions": {
        "Name": "UpperThresholdNonCritical",
        "Value": "48.1",
        "DirectionOfCrossing": "Increasing"
      }
    }
  }
],

```

3.1.3. Specifying Calculations

There are two options for modeling a calculate metrics proposed the telemetry model. The first option is in the `MetricDefinition` resource. The second is in the `MetricReportDefinition` resource.

The first option can be used when a metric property already exists for the result of calculation. For example, the `Power` resource has the `AverageConsumedWatts` property whose value is the calculated value. In this example a `MetricDefinition` exists for `AverageConsumedWatts`. The `MetricDefinition` contains the calculation properties. The `CalculationParameters` property contains a references to the source and resultant metric properties.

```

"CalculationAlgorithm": "AverageOverInterval",
"CalculationTimeInterval": "PT1S",
"CalculationParameters": [
  {
    "SourceMetric": "/redfish/v1/Chassis/...PowerConsumedWatts",
    "ResultMetric": "/redfish/v1/Chassis/...AverageConsumedWatts"
  },
  ...
]

```

The second option can be used when no metric property exists for the result of a calculation. In the case, the calculation can be specified within the `MetricReportDefinition` directly.

```

"Metrics": [
  {
    "MemberID": "AverageConsumedWatts",
    "CollectionFunction": "Avg",
    "MetricProperties": [ "/redfish/v1/Chassis/...PowerConsumedWatts" ]
  }
]

```

```

    },
    ...
]

```

Note: The calculations are specified by name (e.g. `AverageOverInterval`). This works for finite set of commonly used formulas. There is the possibility of adding a mechanism of describing general formulas. This is not currently addressed in the telemetry model.

3.1.4. Specifying Duration

In Redfish, the specification of points-in-time should conform to the ISO 8601 date-time format.

In general, the Redfish convention appends units-of-measure to the name of a property (e.g. `PowerConsumedWatts`). The units-of-measure should conform to the Unified Code for Units of Measure (UCUM).

For durations, the Redfish model uses various methods, as shown in the examples, below:

- `Power#/PowerControl/PowerMetrics/IntervalInMin`
- `Power#/PowerControl/PowerLimit/CorrectionInMs`
- `EventService#/DeliveryRetryIntervalSeconds`
- `Setting#/MaintenanceWindowDurationInSeconds`
- `SessionService#/SessionTimeout`

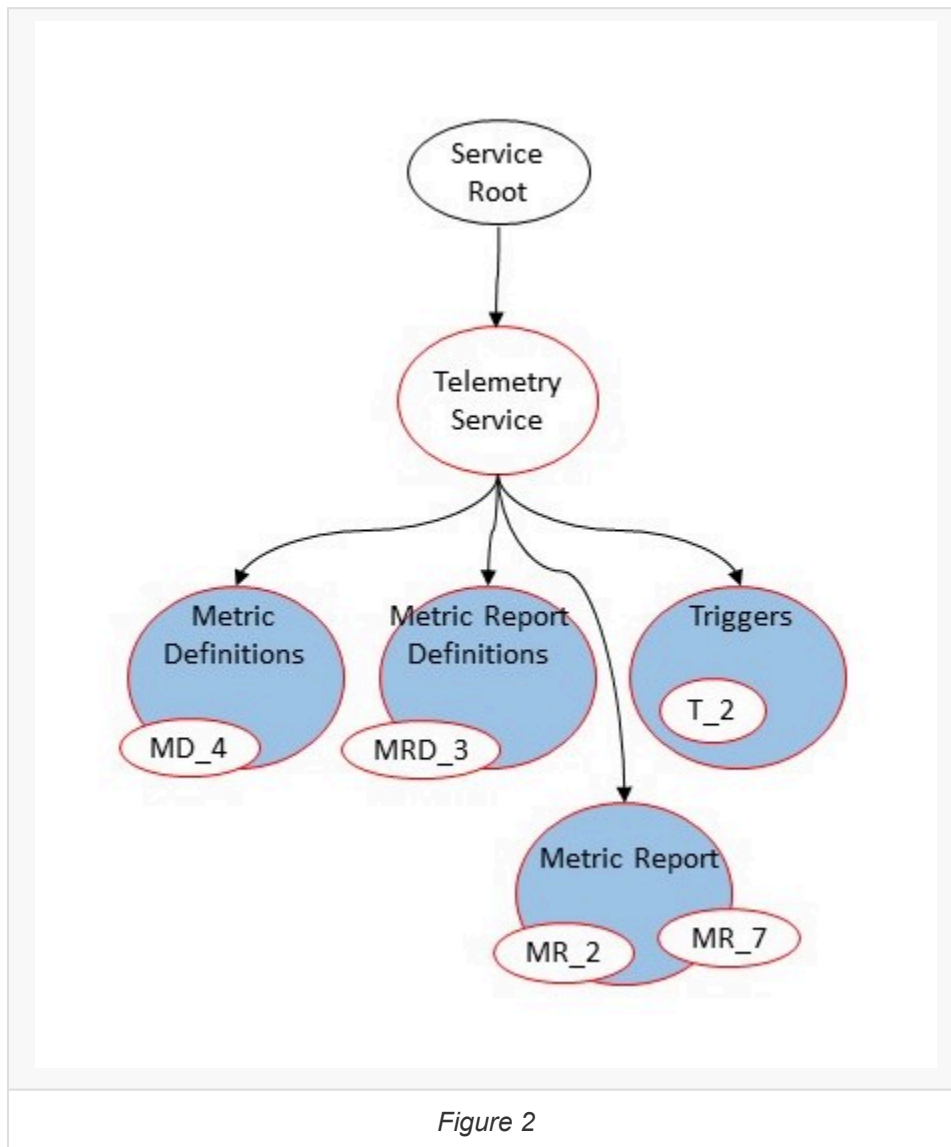
The telemetry model specifies durations using the ISO 8601 duration format.

```
P[n]Y[n]M[n]DT[n]H[n]M[n]S
```

- P is the duration designator (for period) placed at the start of the duration representation
- Y is the year designator that follows the value for the number of years
- M is the month designator that follows the value for the number of months
- W is the week designator that follows the value for the number of weeks
- D is the day designator that follows the value for the number of days
- T is the time designator that precedes the time components of the representation
 - H is the hour designator that follows the value for the number of hours
 - M is the minute designator that follows the value for the number of minutes
 - S is the second designator that follows the value for the number of seconds. This value may have a decimal for values less than a second.

3.2. Telemetry Service

The TelemetryService resource is the top level resource visible on ServiceRoot.



The TelemetryService contains links to collections of MetricDefinitions, MetricReportDefinitions, MetricReports, and Triggers. MetricDefinitions are described in the [Metric Definitions](#) section; MetricReportDefinitions and MetricReports are described in the [Metric Report Definitions](#) section; and, Triggers are described in the [Triggers](#) section.

The telemetry service model is constructed to minimize the impact to the current metric model where metric properties can be place anywhere in the Redfish model. The goal was to not require significant changes in existing metric properties and allow metric characteristics and capabilities to be added

incrementally.

Hence, the TelemetryService is optional. An implementation can decide for each metric property, whether the associated metric definition, metadata or characteristics are provided, and the amount of metadata available.

Furthermore, subordinate resources (MetricDefinition, MetricReportDefinition and Triggers) are independent on each other. So an implementation can support one or all of the subordinate resources and the capabilities represented by them.

Example Telemetry Service Resource:

```
{
  "@odata.context": "/redfish/v1/$metadata#TelemetryService",
  "@odata.type": "#TelemetryService.v1_0_0.TelemetryService",
  "@odata.id": "/redfish/v1/TelemetryService",
  "Id": "TelemetryService",
  "Name": "Telemetry Service",
  "Status": {
    "State": "Enabled",
    "Health": "OK"
  },
  "SupportedCollectionFunctions": ["Avg", "Min", "Max"],
  "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"
  }
}
```

3.3. Metric Definitions

The MetricDefinitions collection resource contains MetricDefinition singleton resources. Each MetricDefinition contains the definition, metadata, or characteristics for a metric. In Figure 3, PowerConsumedWatts is MetricDefinition for the PowerConsumedWatts property in the Power resource.

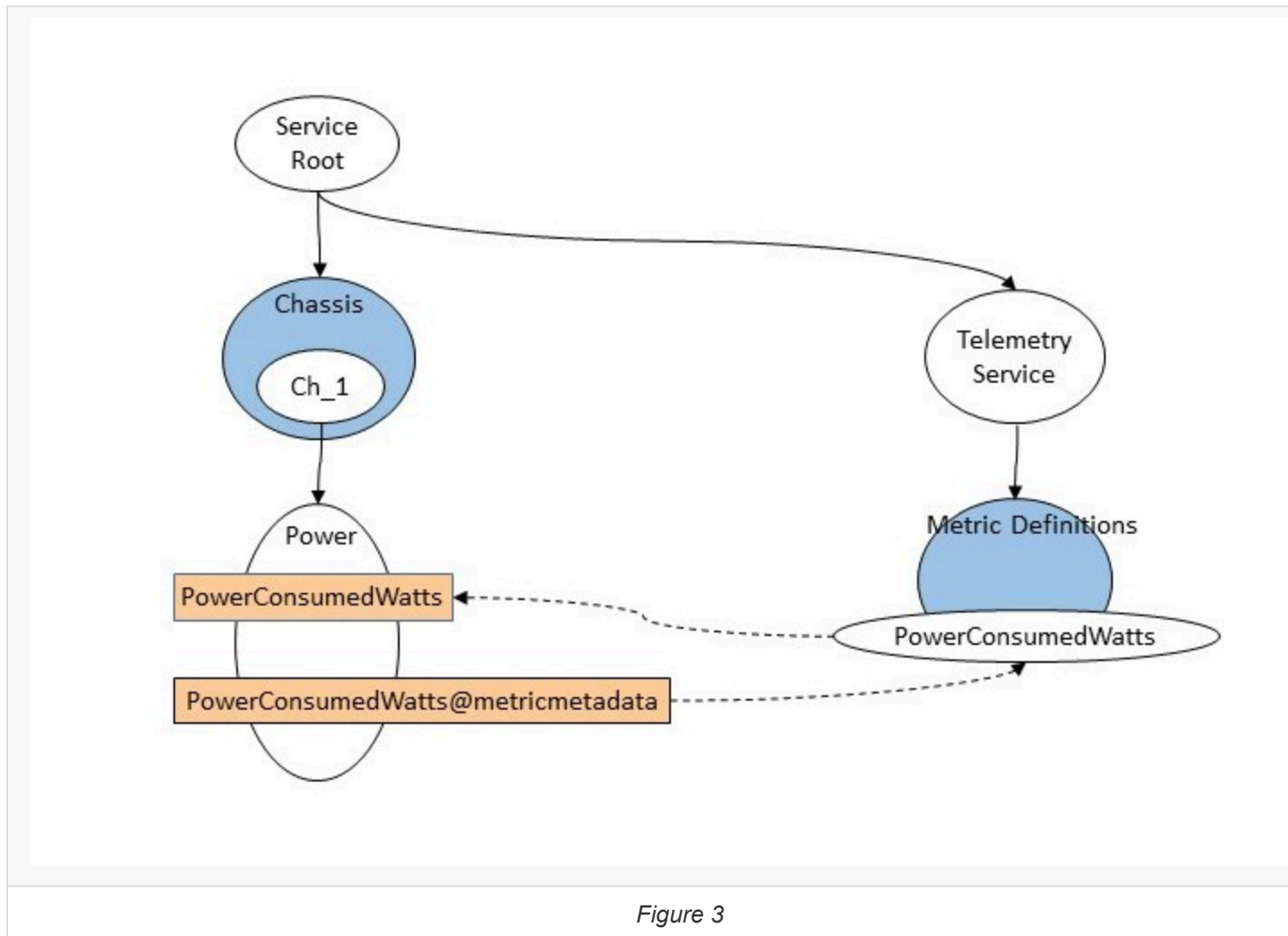


Figure 3

The MetricDefinition resource contains a MetricProperties object which references the metric properties to which the metric definition applies. In Figure 3, the PowerConsumedWatts metric definition can reference each PowerConsumedWatts property in every Chassis, if that represents the implemented Redfish service.

From the PowerConsumedWatts metric property in the Power resource, an annotation may be inserted to reference the MetricDefinition which applies to that metric property.

The characteristics represented in the MetricDefinition resource can be grouped into four categories: context, usage and measurement.

The context properties contain values which provide the context of the metric. The context properties are:

- MetricType
- ImplementationType

- SensorType
- PhysicalContext

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

- Calculable
- IsLinear

The reoccurring measurement property is the `Schedule` property. The property specifies the interval of reoccurring measurement and lifetime of the recurring measurement. The example below specifies a reoccurrence of .001 seconds for 5 days.

The remaining properties are measurement properties which characterizes the measurement, itself. The measurement properties contain properties obtained from the Energy Efficient HPC WG's PowerAPI specification. In their description below, an excerpt for the PowerAPI specification is included. The PowerAPI properties are:

- Precision
- Accuracy
- TimeStampLatency
- TimeStampAccuracy
- TimeWindow
- UpdateRate
- SampleRate
- MeasureMethod

3.3.1. Metric Type

The MetricType property specifies the type of metric and can have the following values.

- **Counter** - The metric is a monotonic counter
- **Gauge** - The metric has values the can increase or decrease
- **Numeric** - The metric is a counter metric
- **Discrete** - The metric is a counter metric

3.3.2. ImplementationType

The ImplementationType property specifies how the metric is obtained and can have the following values.

- **PhysicalSensor** - A physical sensor measures values of the physical environment
- **DigitalMeter** - A digital meter measures values of digital elements (e.g. cache hits)
- **Calculated** - A calculated metric is obtained by applying a calculation on other metrics
- **Synthesized** - A synthesized metric is a calculated metric which represents physical sensor value

3.3.3. SensorType

The SensorType property contains the value from the SensorType enumerated in LogEntry.xml

3.3.4. PhysicalContext

The PhysicalContext property contains the value from the SensorType enumerated in PhysicalContext.xml

3.3.5. Units

The Units property is the units of the metric, as defined by Unified Code for Units of Measure (UCUM)

3.3.6. DiscreteValues

The DiscreteValues array property is a list of the discrete values that a Discrete metric value may take.

3.3.7. Precision

The Precision property contains a value as specified by the Power API.

Number of significant digits in values

3.3.8. Accuracy

The Accuracy property contains a value as specified by the Power API.

Estimated percent error +/- of measured vs. actual values.

3.3.9. TimeStampLatency

The TimeStampe property contains a value as specified by the Power API.

Estimate of the time required to get or set an attribute. This is useful to estimate completion time for an operation a priori. A value of zero should be returned when the get/set is instantaneous.

3.3.10. TimeStampAccuracy

The TimeStampAccuracy property contains a value as specified by the Power API.

Estimated accuracy of returned timestamps, represented as +/- the PWR_Time value returned.

3.3.11. TimeWindow

The TimeStampAccuracy property contains a value as specified by the Power API.

The time window used to calculate the value returned or relevant to an attribute. For example, the "instantaneous" PWR_ATTR_POWER values reported may actually be averaged over a short time window. Power caps are also enforced with respect to a target time window.

3.3.12. UpdateRate

The UpdateRate property contains a value as specified by the Power API.

Rate values become visible to user, in updates per second. Getting or setting a value at a rate higher than this is not useful.

3.3.13. SampleRate

The SampleRate property contains a value as specified by the Power API.

Rate of underlying sampling, in samples per second. This is only relevant for values derived over time (e.g., PWR_ATTR_ENERGY)

3.3.14. MeasureMethod

The MeasureMethod property contains a value as specified by the Power API.

Denotes the measurement method: an actual measurement (returned value = 0) or a model based estimate (return value = 1). Other values > 1 may be used to denote multiple vendor specific models in the situation where multiple models may exist.

3.3.15. Wildcards

The Wildcards property is used on conjunction with the MetricProperties property. The MetricProperties property contains a list of each metric property described by the MetricDefinition. This list could get very large. In order to reduce the size of the list, the MetricProperties strings contain wildcards, delimited by curly braces "{}".

The Wildcards property contains a list of the wildcards. Each wildcard contains a list the values that should be replace the wildcard. A value of "*", would mean all values.

The wild card mechanism is also used in other parts of the Telemetry model.

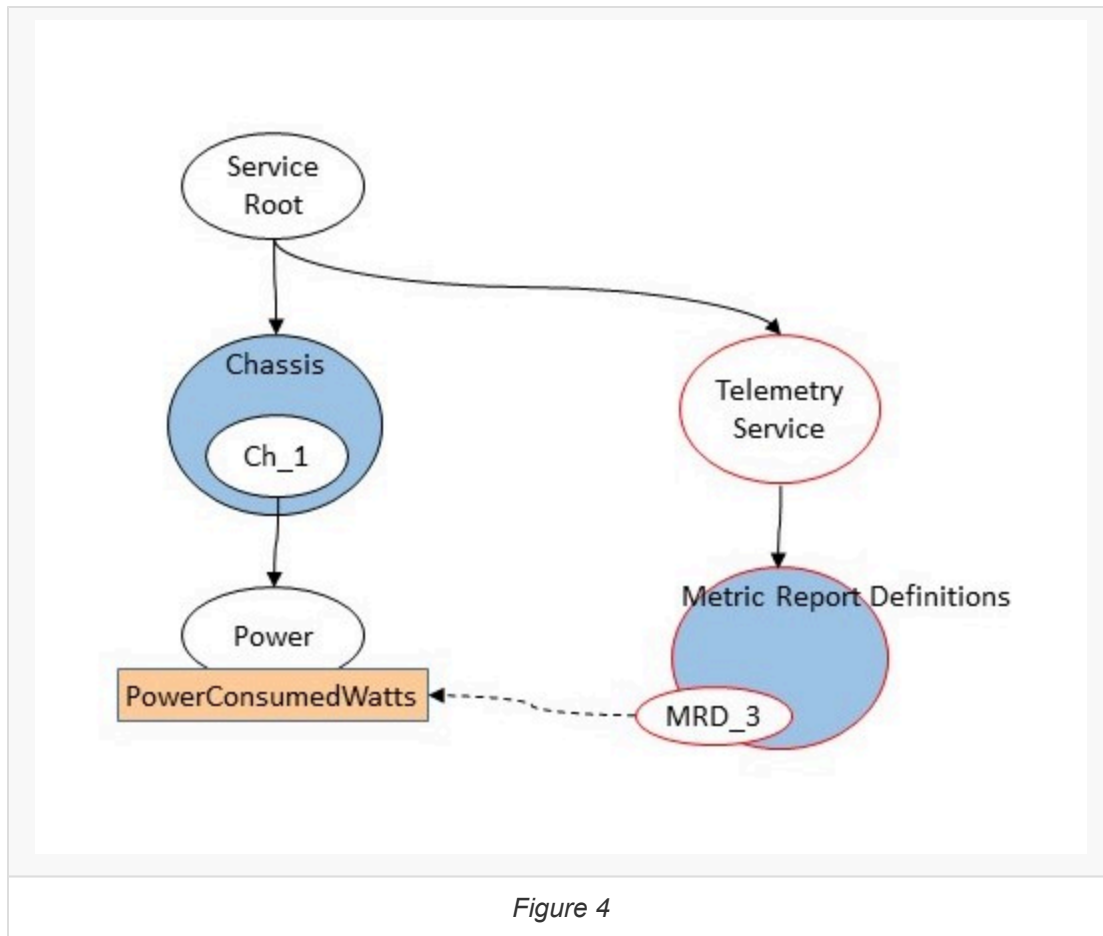
3.3.16. Example

The following example is for a numeric sensor, PowerConsumedWatts properties, which exists in the Power resources.

```
{
  "@odata.context": "/redfish/v1/$metadata#MetricDefinition.MetricDefinition",
  "@odata.type": "#MetricDefinition.v1_0_0.MetricDefinition",
  "@odata.id": "/redfish/v1/TelemetryService/MetricDefinitions/PowerConsumedWatts",
  "Id": "PowerConsumedWatts",
  "Name": "Metric Definition of Power Consumed",
  "MetricType": "Numeric",
  "SensorType": "PowerConsumption",
  "Implementation": "PhysicalSensor",
  "PhysicalContext": "PowerSupply",
  "SensingInterval": "1000",
  "Units": "W",
  "Precision": 4,
  "Accuracy": 1.0,
  "Calibration": 2,
  "TimeStampAccuracy": "PT1S",
  "MinReadingRange": 0.0,
  "MaxReadingRange": 50.0,
  "Wildcards": [
    { "Name": "ChassisID", "Values": [ "1", "2", "3" ] }
  ],
  "MetricProperties": [
    "/redfish/v1/Chassis/{ChassisID}/Power#/PowerControl/0/PowerConsumedWatts",
    "/redfish/v1/Chassis/{ChassisID}/Power#/PowerControl/1/PowerConsumedWatts"
  ]
}
```

3.4. Metric Report Definitions

The `MetricReportDefinition` resource specifies the metric report that the Redfish service will create. The metric reports can be used to aggregate metric readings. The metric reports can be created periodically, when a reading value changes, or upon request. The metric report can be transmitted using the Event Service and/or stored locally (as a member of the `./MetricReports` collection) and retrieved later.



The properties of MetricReportDefinition are described below.

3.4.1. MetricReportType

The `MetricReportType` property specifies when the report is created and can have the following values.

- **Periodic** - The metric report is updated periodically
- **OnChange** - The metric report is updated when the values change
- **OnReport** - The metric report is updated each time a client reads it

3.4.2. ReportActions

The `ReportActions` array property specifies the action(s) to perform when a metric report is generated.

- **Log** - Place the metric report in a location where the client can retrieve
- **Transmit** - Send the metric report as a Metric event

3.4.3. MetricProperties

The `MetricProperties` array property specifies metrics which are metrics are included in the metric report. The `MetricProperties` may have wild cards.

3.4.4. Example

The following example specifies a metric report with includes the `AvgPowerConsumedWatts`, `MinPowerConsumedWatts` and `MaxPowerConsumedWatts`, from the chassis, `Tray_1`, `Tray_2` and `Tray_3`. The metric report is to generated periodically and transmit as a Informational Event and also logged as a member of the `MetricReports` collection resource. When logging, the metric report should overwrite a previous metric report.

```
{
  "@odata.context": "/redfish/
v1/$metadata#MetricReportDefinition.MetricReportDefinition",
  "@odata.type": "#MetricReportDefinition.v1_0_0.MetricReportDefinition",
  "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/
JL_PowerMetrics",
  "Id": "JL_PowerMetrics",
  "Name": "Power Metrics",
  "MetricReportType": "Periodic",
  "Schedule": {
    "RecurrenceInterval": "PT0.1S"
  },
  "ReportActions": [ "Transmit", "Log" ],
  "MetricReport": { "@odata.id": "/redfish/v1/TelemetryService/MetricReports/
PowerMetrics" },
  "Volatile": true,
  "Status": {
    "State": "Enabled"
  },
  "Wildcards": [
    { "PWild": [ "0", "1" ] },
    { "TWild": [ "Tray_1", "Tray_2", "Tray_3" ] }
  ],
  "MetricProperties": [
    "/redfish/v1/
Chassis/{TWild}/Power#/PowerControl/{PWild}/AvgPowerConsumedWatts",
    "/redfish/v1/
Chassis/{TWild}/Power#/PowerControl/{PWild}/MinPowerConsumedWatts",
    "/redfish/v1/Chassis/{TWild}/Power#/PowerControl/{PWild}/MaxPowerConsumedWatts"
  ]
}
```

3.5. Triggers

The Triggers resource specifies the trigger threshold(s) that apply to numeric or discrete metrics. A trigger can result in an alert being transmitted using the Event Service and/or logged in the service log.

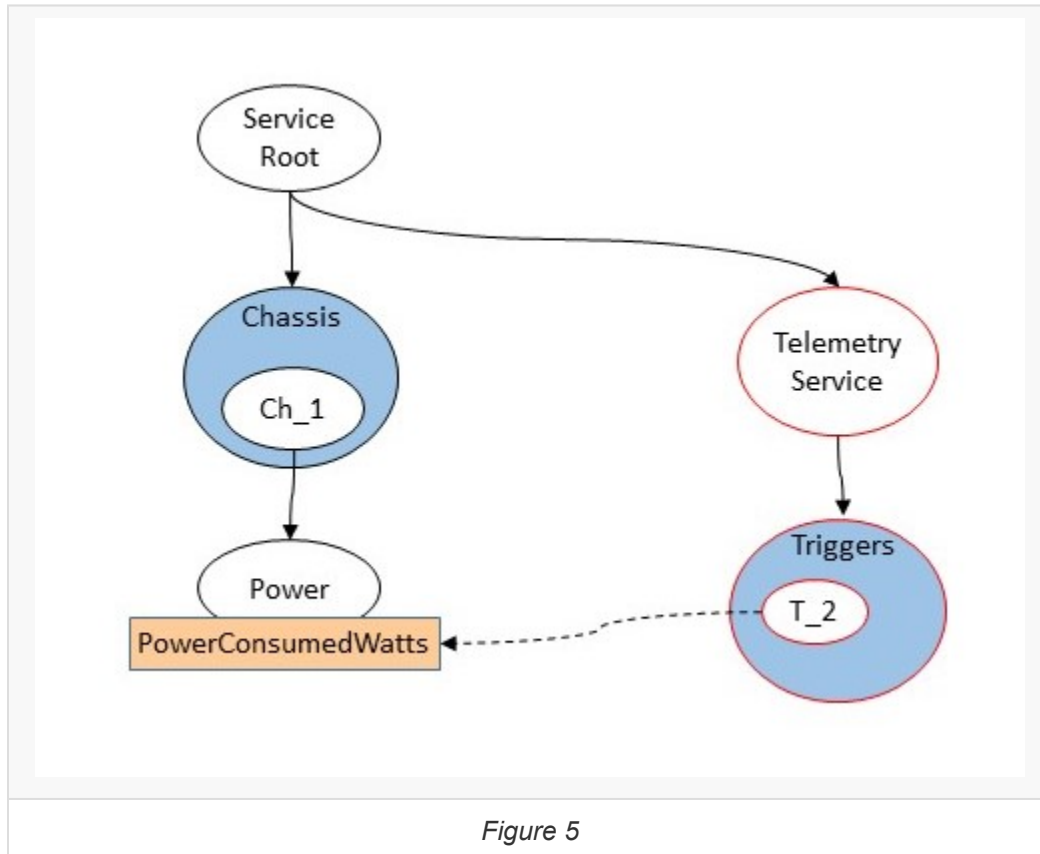


Figure 5

3.5.1. TriggerType

The `TriggerType` property specifies the type of trigger and indicates other properties that should be present. The property can have the following values:

- **Numeric** - The triggers are numeric. See `NumericTriggers` property
- **Discrete** - The triggers are discrete. See `DiscreteTriggerCondition` property

3.5.2. TriggerActions

The `TriggerActions` array property specifies the action(s) to perform when a trigger is trip. The property can have the following values:

- **Log** - Log the trigger into the Service Log

- **Transmit** - Send the trigger as a Alert event

3.5.3. NumericTriggers

The `NumericTriggers` array property specifies the trigger thresholds, if the `TriggerType` is numeric. The property is complex and has the following properties

- `Value` - the value of the threshold
- `DirectionOfCrossing` - the direction that the threshold is crossed to trip the trigger
- `DwellTimems` - the duration that the trigger is tripped before the trigger action is invoked
- `Severity` - the value of the `Severity` property within the alert Event Message

3.5.4. DiscreteTriggerConditions

The `DiscreteTriggerCondition`

- **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.

3.5.5. DiscreteTriggers

The `DiscreteTriggers` array property specifies the values of metric which will trip the trigger, if the `TriggerType` is discrete and `DiscreteTriggerCondition` is specified.

- `Value` - the value of the threshold
- `DwellTimems` - the duration that the trigger has tripped before the trigger action is invoked
- `Severity` - The value of the `Severity` property within the alert Event Message

3.5.6. MetricProperties

The `MetricProperties` array property specifies metrics which are metrics to monitor against the triggers. The `MetricProperties` property may have wildcards.

3.5.7. Example

The following is an example of a numeric trigger with two thresholds (Upper Threshold Critical and Non Critical). When the trigger occurs, an Alert Event is sent to the subscribers, and will include the corresponding severity.

```
{
  "@odata.context": "/redfish/v1/$metadata#Triggers.Triggers",
```

```

"@odata.type": "#Triggers.v1_0_0.Triggers",
"@odata.id": "/redfish/v1/TelemetryService/Triggers/PlatformPowerCapTriggers",
"Id": "PlatformPowerCapTriggers",
"Name": "Triggers for platform power consumed",
"MetricType": "Numeric",
"TriggerActions": [ "Transmit" ],
"NumericTriggers": [
  {
    "Name": "UpperThresholdCritical",
    "Value": "50.0",
    "DirectionOfCrossing": "Increasing",
    "DwellTimems": "1",
    "Severity": "Critical"
  },
  {
    "Name": "UpperThresholdNonCritical",
    "Value": "48.1",
    "DirectionOfCrossing": "Increasing",
    "DwellTimems": "4",
    "Severity": "Warning"
  }
],
"MetricProperties": [
  "/redfish/v1/Chassis/1/Power#/PowerControl/0/PowerConsumedWatts",
  "/redfish/v1/Chassis/1/Power#/PowerControl/1/PowerConsumedWatts"
]
}

```

4. References

- [Redfish Telemetry Model Proposal](http://www.dmtf.org/sites/default/files/standards/documents/DSP-IS0002_0.9a.zip), Distributed Management Task Force, http://www.dmtf.org/sites/default/files/standards/documents/DSP-IS0002_0.9a.zip
- [PowerAPI Specification](http://www.ietf.org/rfc/rfc3986.txt), Energy Efficient HPC Working Group, <http://www.ietf.org/rfc/rfc3986.txt>
- [The Unified Code for Units of Measure](http://www.unitsofmeasure.org/ucum.html), Unified Code for Units of Measure (UCUM) Organization, <http://www.unitsofmeasure.org/ucum.html>
- [Date and time format - ISO 8601](https://www.iso.org/standard/40874.html), International Organization for Standardization, <https://www.iso.org/standard/40874.html>