White Paper

Telemetry Streaming with iDRAC9— Custom Reports Get Started

Abstract

Dell EMC PowerEdge Servers with iDRAC9 4.0 Datacenter stream data to help IT administrators better understand the inner workings of their server environment. This white paper explains the Telemetry Streaming feature and basic steps to configure iDRAC9, including adding custom report definitions on iDRAC9 4.40 or above.

May 2021

Document 365/385

White Paper

Revisions

Date	Description
May 2021	Initial release

Acknowledgments

Authors: Sankara Gara, Sailaja Mahendrakar, Heidi Maeder, Praveen Thangavelu, Doug Iler

Table of contents

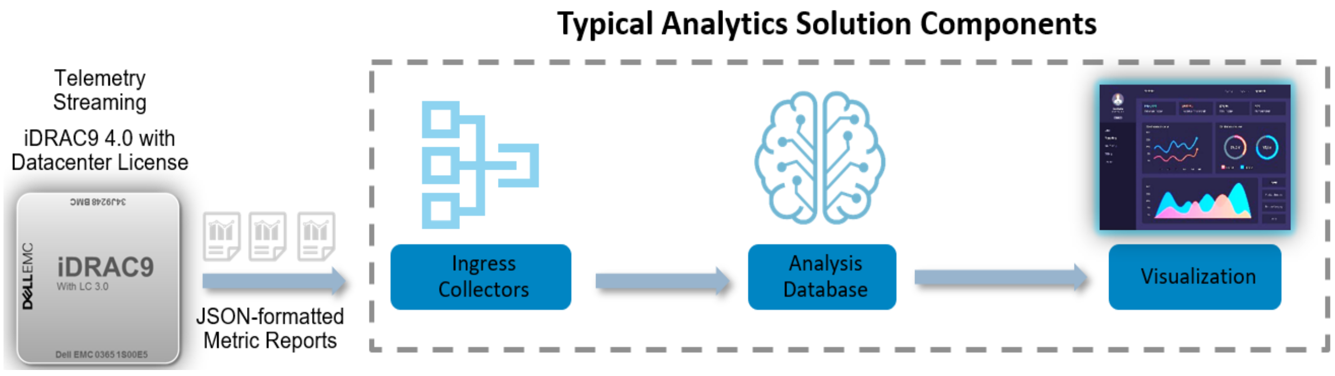
Executive summary

With iDRAC9 v4.00.00.00 firmware and the Datacenter license, IT managers can integrate advanced server hardware operation telemetry into their existing analytics solutions. Telemetry is provided as granular, time-series data that is streamed, or pushed, compared to inefficient, legacy polling, or pulled, methods. The advanced agent-free architecture in iDRAC9 provides over 180 data metrics that are related to server and peripherals operations. Metrics are precisely timestamped and internally buffered to allow highly efficient data stream collection and processing with minimal network loading. This comprehensive telemetry can be fed into analytics tools to predict failure events, optimize server operation, and enhance cyber resiliency.

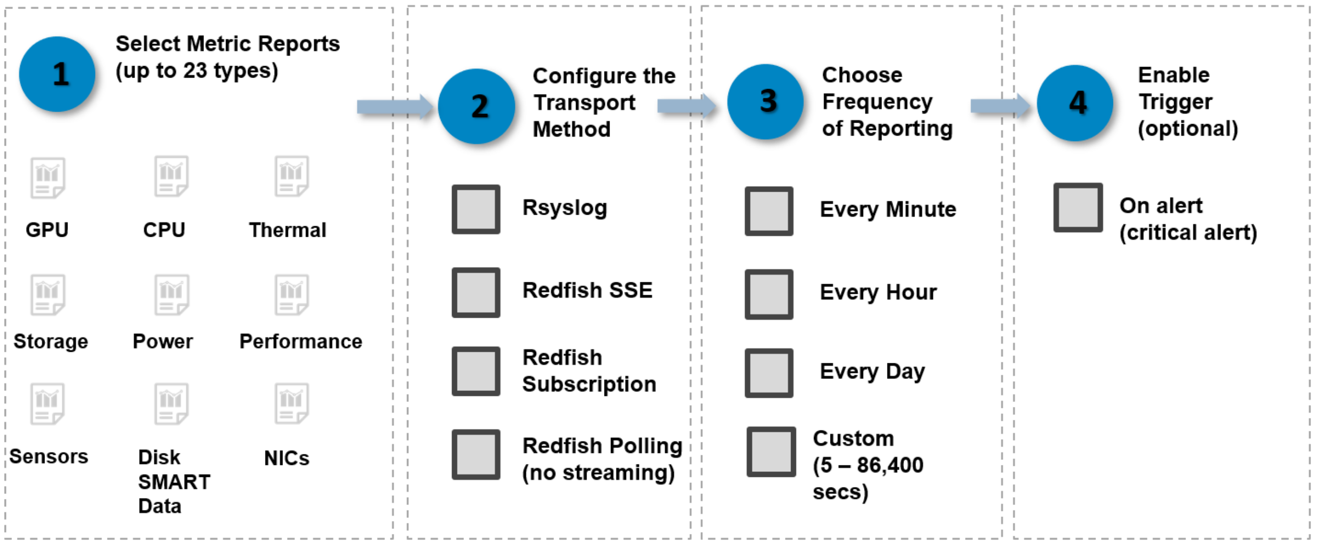
1 Telemetry Overview

Telemetry streaming is an automated communications process by which measurements and other data are collected at remote or inaccessible points. With iDRAC9 4.0 Datacenter, it is possible to stream a wide variety of metric reports from to an ingress collector such as Splunk or ELK Stack. These and other tools can then perform remote server monitoring and analysis.

The following diagram shows the basic elements used for Telemetry Streaming Analytics


Typical Analytics Solution Components
Telemetry Streaming
iDRAC9 4.0 with Datacenter License
iDRAC9
With LC 3.0
DELL EMC
JSON-formatted Metric Reports
Ingress Collectors
Analysis Database
Visualization

This paper will focus on the items under iDRAC control, as shown below:


1
Select Metric Reports (up to 23 types)
GPU
CPU
Thermal
Storage
Power
Performance
Sensors
Disk SMART Data
NICs
2
Configure the Transport Method
Rsyslog
Redfish SSE
Redfish Subscription
Redfish Polling (no streaming)
3
Choose Frequency of Reporting
Every Minute
Every Hour
Every Day
Custom (5 – 86,400 secs)
4
Enable Trigger (optional)
On alert (critical alert)

1.1 Terms and Definitions

Telemetry Report: A telemetry report is a DMTF Redfish Telemetry Specification compliant JSON document that consists of metric names, metric values, and timestamps.

SSE: Server-sent events allow for a client to open a web service connection which can continuously push data to the client as needed.

EEMI: The *Event and Error Message Information* is available in a reference guide which lists the messages in the user interface, command-line interface, and log files. Messages are displayed or stored as a result of user action, automatic event occurrence, or for data logging purposes.

MRD – Metric Report Definition

FQDD – Fully Qualified Device Descriptor

1.2 Prerequisites

The Custom Telemetry feature is available on iDRAC9 firmware version 4.40.00.00 or above and requires Datacenter license.

2 Configuring Basic Telemetry

Telemetry configuration allows you to configure telemetry data streaming content and behavior. It includes settings to enable Telemetry Service and settings specific to each available report. Enabling or disabling Telemetry Service enables or disables telemetry streaming for all reports. By default, the Telemetry Service and all pre-canned reports are disabled. Once enabled, the telemetry reports are sent to connected Redfish clients using the HTTPS protocol.

The Telemetry Service details can be obtained from HTTP GET of top-level Telemetry URI – */redfish/v1/TelemetryService*. The details include the current service status and the supported URIs and OEM Dell specific information such as currently active FQDDs and Sources for the metric values.

The basic steps involved in getting started with telemetry steaming with the included pre-canned reports are:

1. Enabling telemetry service
2. Configuring report streaming content and behavior
3. Configuring report triggers (optional)
4. Subscribing to the telemetry reports
5. Receiving Telemetry reports (outside iDRAC).

For creating custom reports with desired metrics and other advanced telemetry streaming behavior setting please refer to section 2.5

2.1 Enable Telemetry Service

To view current Telemetry Service state enter following HTTP command.

```
curl -s -k -u <user>:<password> -X GET https://<IDRAC-IP>//redfish/v1/TelemetryService
          -H 'Content-Type: application/json'
```

To enable Telemetry Service:

```
curl -s -k -u <user>:<password> -X PATCH https://<IDRAC-IP>//redfish/v1/TelemetryService
```

```
          -H 'Content-Type: application/json' -d '{"ServiceEnabled": true }'
```

To disable Telemetry Service:

```
 curl -s -k -u <user>:<password> -X PATCH https://<IDRAC-IP>//redfish/v1/TelemetryService
          -H 'Content-Type: application/json' -d '{"ServiceEnabled": false }'
```

2.2 Configuring Telemetry Streaming Content

The system is shipped with pre-canned report definitions with default configuration for periodic reporting. At the minimum, the desired reports should be enabled to stream the reports at preconfigured recurrence interval. For customizing the pre-canned reports or adding new custom reports please refer to the section below.

To view currently available pre-canned report definition collection:

```
 curl -s -k -u <user>:<password> -X GET https://<IDRAC-
IP>//redfish/v1/TelemetryService/MetricReportDefinitions
          -H 'Content-Type: application/json'
```

To view the details of a report definition:

```
 curl -s -k -u <user>:<password> -X GET https://<IDRAC-
IP>//redfish/v1/TelemetryService/MetricReportDefinitions/<report>
          -H 'Content-Type: application/json'
```

To enable a report:

```
 curl -s -k -u <user>:<password> -X PATCH https://<IDRAC-
IP>//redfish/v1/TelemetryService/MetricReportDefinitions/<report>
          -H 'Content-Type: application/json' -d '{" MetricReportDefinitionEnabled": true}'
```

To configure report recurrence interval if different from default (e.g. 2 minutes):

```
 curl -s -k -u <user>:<password> -X PATCH https://<IDRAC-
IP>//redfish/v1/TelemetryService/MetricReportDefinitions/<report>
          -H 'Content-Type: application/json' -d '{ "Schedule":{"RecurrenceInterval":
"PT0H2M0S"}}'
```

To disable a report:

```
 curl -s -k -u <user>:<password> -X PATCH https://<IDRAC-IP>//redfish/v1/TelemetryService/
MetricReportDefinitions/<report>
          -H 'Content-Type: application/json' -d '{" MetricReportDefinitionEnabled": false}'

e.g.  <report> = PowerMetrics
```

2.3 Configuring Telemetry Report Triggers (optional)

Telemetry triggers are a means to generate and stream reports that are based on an error or warning condition. These reports are predefined based on Lifecycle log (LCL) events for error or warming conditions. If configured, a new report is generated before the scheduled report interval when a trigger occurs. The default configuration includes the triggers that are relevant for a report. You can modify the trigger association.

```
HTTP PATCH /redfish/v1/Managers/iDRAC.Embedded.1/Attributes
Payload: {"Attributes":{"Telemetry<report>.1.ReportTriggers": “<trig1, trig2>"}

e.g.
curl -s -k -u <user>:<password> -X PATCH https://<IDRAC-
IP>/redfish/v1/Managers/iDRAC.Embedded.1/Attributes
-H 'Content-Type: application/json' -d '{"Attributes":
{"TelemetryPowerMetrics.1.ReportTriggers": "CPUCriticalTrigger, CPUWarnTrigger"}}'
```

2.4 Receiving Telemetry Reports

After telemetry streaming is configured, the telemetry reports can be received by a Redfish client using these methods. First two are streaming and the last is pull a report on demand.

1. POST to Subscription method
2. SSE Method
3. Pull (GET) Method

2.4.1 POST to Subscription Method

In this method Redfish clients first create subscription(s) using destination (ip:port) and desired reports in the subscriptions request. If no report list is specified then all enabled reports will be streamed. Then the clients start HTTP event listener on the destination that listens on the port to receive the telemetry report streams periodically, as configured above. Maximum 8 subscriptions, including internal SSE subscriptions (see SSE method), can be created.

To create a subscription:

```
 curl -s -k -u <user>:<password> -X POST https://<IDRAC-
IP>/redfish/v1/EventService/Subscriptions

            -H 'Content-Type: application/json'

           -d '{"Destination": "https://<listener ip:port>",

                  "EventFormatType": "MetricReport",

                  "Context": "TelmetryTest",

                  "Protocol": "Redfish",
                  "EventTypes": ["MetricReport"],

                  "SubscriptionType":"RedfishEvent",

                  "MetricReportDefinitions":[

                  {

                         "@odata.id":
                  "/redfish/v1/TelemetryService/MetricReportDefinitions/<report-1>",

                         "@odata.id":
                  "/redfish/v1/TelemetryService/MetricReportDefinitions/<report-2>"

                  }

                  ]

           }'
```

View current subscription collection:

```
curl -s -k -u <user>:<password> -X GET https://<IDRAC-
IP>/redfish/v1/EventService/Subscriptions

           -H 'Content-Type: application/json'
```

Delete a subscription:

```
curl -s -k -u <user>:<password> -X DELETE https://<IDRAC-
IP>/redfish/v1/EventService/Subscriptions/<subscrition-id>

           -H 'Content-Type: application/json'
```

2.4.2 SSE Method

In this method Redfish client simply invokes HTTP GET on SSE URI with EventFormat type as “Metric Report”. This establishes a connection between the client and iDRAC Telemetry Service, and all enabled reports are streamed periodically, as configured above. This also adds an internal client subscription which gets deleted when the connection is closed. The connection can be terminated by either the client or iDRAC Telemetry service. If there is no telemetry data being sent to client for more than an hour, the connection gets terminated from the service endpoint. If there is a connection issue due to a network glitch or for unknown reasons, then the last event id is presented by the client to the service to resume streaming.

Streaming all reports:

```
curl -N -k - u <user>:<password> -X GET 'https://<IDRAC-
IP>/redfish/v1/SSE?$filter=EventFormatType%20eq%20MetricReport'
```

Streaming a single report:

```
curl -N -k -u <user>:<password> -X GET 'https://<IDRAC-
IP>/redfish/v1/SSE?$filter=MetricReportDefinition%20eq%20%27/redfish/v1/TelemetryService/
MetricReportDefinitions/<report>%27'
```

2.4.3 Pull Method

The Redfish client can pull a report or report collection URIs on demand by performing an HTTP GET operation on the metric report URI as specified below.

Pull report collection (URI list only) to know available reports:

```
curl -s -k -u <user>:<password> -X GET https://<IDRAC-
IP>/redfish/v1/TelemetryService/MetricReports
```

Pull one report:

```
curl -s -k -u <user>:<password> -X GET https://<IDRAC-
IP>/redfish/v1/TelemetryService/MetricReports/<report>

e.g. <report> = PowerMetrics
```

2.5 Configuring Custom Telemetry Reports

Telemetry streaming solution includes a large list of metrics with metric report definition that contains a list of properties. Not all users might be interested in all the metric and properties. There comes the need to configure the MRD with a custom set of metrics and properties . Each client/user can control the properties of its named report, specifying recurrence interval, report type, aggregation, etc. independently. And the metric reports can be customized by selecting the needed arbitrary metrics in the metric report definition. The existing pre-canned reports can be also modified or configured with desired metrics and properties, but that would impact all clients that use the pre-canned report. For the detailed explanation of the all available MRD properties please refer to the white paper “Metric Report Definition Explained”

2.5.1 Create New Custom Report

Below is an example to post a custom MRD with specific properties using Redfish interface. Typically, one can GET any existing pre-canned report definition (MRD) and update the metrics and properties (at the minimum different “Id” value should be specified) and POST the updated json as shown in the example below where a custom report for NIC Tx and Rx bytes metrics for a desired NIC port (FQDD) - *NIC.Slot.1-1*-1 is requested.

```
curl -s -k -u <user>:<password> -X POST https://<IDRAC-
IP>/redfish/v1/TelemetryService/MetricReportDefinitions

           -H 'Content-Type: application/json'

          -d ‘{

              "Id": "TxRxBytesNicSlot1",
              "Name": "Tx and Rx Bytes from Nic Slot 1 Metric Report",
              "Description": "Tx and Rx Bytes of Nic Slot1 record",
              "MetricReportDefinitionEnabled": true,
              "MetricReportDefinitionType": "Periodic",
              "MetricReportHeartbeatInterval": "PT0H0M0S",
              "SuppressRepeatedMetricValue": false,
              "ReportTimespan": "PT0H0M0S",
              "ReportUpdates": "Overwrite",
              "ReportActions": [
                  "RedfishEvent"
              ],
              "Schedule": {
                  "RecurrenceInterval": "PT0H2M0S"
              },
               "Metrics": [

            {
                "MetricId": "TxBytes",
                "MetricProperties": [],
                "MetricProperties@odata.count": 0,
                "CollectionFunction": null,
                "CollectionDuration": null,
                "CollectionTimeScope": "Point",
                "Oem": {
                    "Dell": {
                        "@odata.type": "#DellMetric.v1_1_0.DellMetric",
                        "CustomLabel": null,
                        "FQDD": "NIC.Slot.1-1-1",
                        "Source": null
                   }
               }
           },
           {
               "MetricId": "RxBytes",
               "MetricProperties": [],
               "MetricProperties@odata.count": 0,
               "CollectionFunction": null,
               "CollectionDuration": null,
               "CollectionTimeScope": "Point",
               "Oem": {
                   "Dell": {
                        "@odata.type": "#DellMetric.v1_1_0.DellMetric",
                        "CustomLabel": null,
                        "FQDD": "NIC.Slot.1-1-1",
                        "Source": null
                   }
               }
           }
         ],
          "Metrics@odata.count": 2,
              "Links": {
```

```
                    "Triggers": []
                }
            }
```

When the above POST command successful the new custom report will be added to the report definition collection.

To get report definition collection (URI list only):

```
curl -s -k -u <user>:<password> -X GET https://<IDRAC-
IP>/redfish/v1/TelemetryService/MetricReportDefinitions
```

To get one report definition detail:

```
curl -s -k -u <user>:<password> -X GET https://<IDRAC-
IP>/redfish/v1/TelemetryService/MetricReports/<MRD>
```

*e.g. <**MRD**> = **TxRxBytesNicSlot1***

2.5.2 Update and Remove Custom Reports

The custom report definitions can be updated or removed as needed.

To update any MRD property that is not read-only.

```
    curl -s -k -u <user>:<password> -X PATCH https://<IDRAC-
IP>//redfish/v1/TelemetryService/MetricReportDefinitions/<MRD>
            -H 'Content-Type: application/json'
               -d ' { "Schedule":{"RecurrenceInterval": "PT0H5M0S"},
                      MetricReportDefinitionType": "OnRequest"}'
```

The example here tries to change `RecurrenceInterval` and `MetricReportDefinitionType` MRD properties.

To delete a report definition:

```
    curl -s -k -u <user>:<password> -X DELETE https://<IDRAC-
IP>//redfish/v1/TelemetryService/MetricReportDefinitions/<MRD>
```

*e.g. <MRD> = **TxRxBytesNicSlot1***

Note: If pre-canned reports are customized (properties changed from default) they can be restored to default values by deleting the report definition (MRD), e.g *PowerMetrics.*

2.6 Report Generation Behavior and Limitations

Metric reports are generated and values are added to the report at the rate they are produced by the backend services. Reports that contain metrics with different reporting characteristics will have different numbers of metric values in the resulting reports that match the rate at which the backend daemons report data for these metrics.

There may a variance in Metric Value count variance because the rate at which metrics are ingested is clocked by backend services reporting the metrics. There is some expected variance in the number of metric values that appear in a report above what you might expect purely by doing the math of "*RecurrenceInterval*" / "*SensorInterval*". Thus, a one minute report that has a metric with a five second sensor interval will not necessarily have exactly 12 entries.

The Metric Reports will not return any metric that has a NULL or invalid value regardless of the suppression or heartbeat implementation. Periodic reports configured with no suppress repeated metrics option shall stream last read data if the server is powered off.

Custom reports are limited to a total of 48 total report definitions, including 24 pre-canned report definitions and potentially 24 custom report definitions.

Reports with a specific configuration**:**

- *NVMeSMARTData* - *NVMeSMARTData* is only supported for SSD (PCIeSSD/NVMe Express) drives with PCIe bus protocol (not behind SWRAID).
- *StorageDiskSMARTData* report is only supported for SSD drives with SAS/SATA bus protocol and behind the BOSS controller.
- *StorageSensor* report is only supported for the drives in non-raid mode and not behind the BOSS controller.
- *GPGPUStatistics* report is only available in specific GPGPU models that support ECC memory capability (GP102GL [Tesla P40]).
- *FanSensor* report gets generated only for Monolithic servers. For modular servers, the report is empty (with "MetricValues@odata.count": 0).
- When server is powered off only these sensor readings are available*: PSU Temperature*, *System Board Inlet,* and *Exhaust Temperatures* on monolithic servers; only *System Board Inlet and Exhaust* temperatures on modular servers*.*
- When a report is enabled but the device hardware is not present, no report is generated. For instance, if a GPU card is not present in the system and the GPUMetrics report is pulled, the result would be an empty report with "*MetricValues@odata.count": 0*.
- When report *RecurrenceInterval* set to 0s, the report can only be pulled and it cannot be streamed. The report will be single instance and non-repeating data, if available at that time of pull.
- If some metrics common across multiple report definitions and *SuppressRepeatedMetricValue* set to true and *MetricReportHeartbeatInterval* set to either less than or greater than the *ReportTimeSpan* on any report then the metric suppression behavior changes and the common metrics are not suppressed.
- If a custom report created with metrics with different sensing intervals, the report would contain only the metrics with lower sensing interval, for example setting the *RecurrenceInterval* less than the lower sensing interval of the metrics.

2.7 Troubleshooting and Tips

Issues	Possible Causes	Solution
POST, PATCH operations failure.	Service is not enabled. Property “ServiceEnabled” is set to false. Applies to TelemetryService/EventService	a) Set Service property “ServiceEnabled” to true. b) Check LC logs
	Property that is added in the input payload is not allowed or the value added is invalid .	a) Check Redfish documentation for allowed properties and valid values.
	User account without “Administrator” privilege	a) Ensure that user account has "Administrator" privilege b) Check LC logs.
DELETE operations failure.	Service is not enabled. Property “ServiceEnabled” is set to false. Applies to TelemetryService/EventService	a) Set Service property “ServiceEnabled” to true. b) Check LC logs
GET Metric Report failure .	Required license is not installed or installed datacenter license is expired	a) Install a new license. b) Check LC logs.
	Service is not enabled. Property “ServiceEnabled” is set to false. Applies to TelemetryService/EventService	a) Set Service property “ServiceEnabled” to true. b) Check LC logs
No Metric Report in the SSE or subscription stream	Service is not enabled. Property “ServiceEnabled” is set to false. Applies to TelemetryService/EventService	a) Set Service property “ServiceEnabled” to true. b) Check LC logs
	Metric report definition (MRD) is not enabled	a) Set MRD property "MetricReportDefinitionEnabled“ to true. b) Check LC logs.
	MRD property "ReportActions" does not have "RedfishEvent"	a) PATCH MRD to have RedfishEvent within ReportActions
	Required license is not installed or installed datacenter license is expired	a) Install a new license. b) Check LC logs.
	No network route to client.	a) Run ping test from iDRAC troubleshooting. b) Get help from infrastructure network administrator.
	Client firewall blocking the port.	a) Check client system firewall and allow port. b) Check LC logs.
Metric Report associated with MRD is empty.	Metric report definition (MRD) is not enabled	a) Set MRD property "MetricReportDefinitionEnabled“ to true. b) Check LC logs.
	MRD property "ReportActions" does not have "RedfishEvent"	a) Set MRD property "ReportActions" to include "RedfishEvent"

2.8

2.8 Best practices

1. ~~A~~ Server Configuration Profile (SCP) is better option to configure all the metric reports by including “Custom Telemetry” option. Once an SCP file is created, the same file can be applied to multiple servers that support Telemetry feature and Datacenter license.

2. Configure the “report interval” based on the system configuration and number of configured telemetry reports. On a max config system, a high report interval (2hr) can in-turn result in large telemetry reports since it includes every relevant device metric. Also, a minimum report interval (10s) can in-turn contribute to processing overheads based on the number active configured reports.

    a. For servers with max configurations (large number of hard drives and or memory cards) it is recommended to not set the *RecurrenceInterval* to maximum value.

    b. For reports like *SystemUsage, PowerMetrics, CPUMemMetrics, ThermalMetrics, and GPUMetrics*, it is recommended to set a minimum *RecurrenceInterval* of 60s even though the minimum *ReportInterval* of 10s is allowed.

2.9 Telemetry Error Messages

Telemetry provides error messaging from 2 sources:

- Dell MessageRegistry – prefixed with (IDRAC.)
- Redfish Specification Base MessageRegistry prefixed with (Base.)
    - Link to Registry: https://redfish.dmtf.org/registries/Base.1.7.0.json

For ease, iDRAC v4.40.00.00 Telemetry Messages are provided below.

Message ID	MessageRegistry Location	Message
Success	DMTF BaseRegistry	Successfully Completed Request
Created	DMTF BaseRegistry	The resource has been created successfully
MalformedJSON	DMTF BaseRegistry	The request body submitted was malformed JSON and could not be parsed by the receiving service.
GeneralError	DMTF BaseRegistry	A general error has occurred. See ExtendedInfo for more information
ResourceMissingAtURI	DMTF BaseRegistry	The resource at the URI <URI> was not found."
ResourceNotFound	DMTF BaseRegistry	The requested resource of type <type> named '<prop>' was not found.
SWC0242	Dell MessageRegistry	A required license is missing or expired. Obtain an appropriate license and try again, or contact your service provider for additional details.
SWC0283	Dell MessageRegistry	The specified object value is not valid.
SYS402	Dell MessageRegistry	The method cannot be run because the requested HTTP method is not allowed.
SYS403	Dell MessageRegistry	Unable to complete the operation because the resource {} entered is not found

SYS406	Dell MessageRegistry	Unable to start the configuration operation because the System Lockdown mode is enabled.
SYS413	Dell MessageRegistry	The operation successfully completed.
SYS414	Dell MessageRegistry	A new resource is successfully created.
SYS419	Dell MessageRegistry	Unable to complete the operation because the Redfish attribute is disabled.
SYS425	Dell MessageRegistry	Unable to complete the operation because the value <value> entered for the property <prop> is invalid.
SYS428	Dell MessageRegistry	Unable to complete the operation because the property
SYS460	Dell MessageRegistry	Unable to perform the necessary telemetry operation because the Telemetry feature is disabled.
SYS479	Dell MessageRegistry	There are insufficient privileges for the account or credentials associated with the current session to perform the requested operation.
SYS482	Dell MessageRegistry	Unable to complete the operation because the MetricReportDefinition <mrd> already exists.
SYS484	Dell MessageRegistry	Unable to complete the operation because <prop> bounds are <value1> to <value2>
SYS485	Dell MessageRegistry	Unable to complete the operation because MetricReportHeartBeatInterval range from the value in the RecurrenceInterval to 24 hours.
SYS488	Dell MessageRegistry	Unable to complete the operation because MetricReportHeartbeatInterval requires to be the value of RecurrenceInterval or greater and SuppressRepeatedMetricValue must be true.
SYS489	Dell MessageRegistry	Unable to complete the operation because the MetricReportDefinitionType, OnChange requires the following: clearing RecurrenceInterval and MetricReportHeartbeatInterval, setting SuppressRepeatedMetricValue to true. ReportTimeSpan to {} or greater.
SYS490	Dell MessageRegistry	Unable to complete the operation because the MetricReportDefinitionType, OnRequest requires RecurrenceInterval to be clear and ReportTimeSpan to be {} or greater.
SYS491	Dell MessageRegistry	Unable to complete the operation because the CollectionDuration and CollectionFunction must be set at the same time. CollectionDuration shall be {} or greater
SYS494	Dell MessageRegistry	The request failed due to an internal service error. The service is still operational.
SYS495	Dell MessageRegistry	The {} was Disabled because the property <prop> was cleared.

A Technical support and resources

- iDRAC Telemetry Workflow Examples
  https://github.com/dell/iDRAC-Telemetry-Scripting/
- Open-source iDRAC REST API with Redfish Python and PowerShell examples.

  https://github.com/dell/iDRAC-Redfish-Scripting
- The iDRAC support home page provides access to product documents, technical white papers, how-to videos, and more.

  www.dell.com/support/idrac
- iDRAC User Guides and other manuals

  www.dell.com/idracmanuals
- Dell Technical Support
  Dell.com/support

B MetricIDs

Following are the currently available metrics (MetricIDs) and the associated pre-canned reports. Detail of each metric (MetricDefinition), like description, type, units, and sensing interval etc., can be obtained using the following command.

```
curl -s -k -u <user>:<password> -X GET https://<IDRAC-
IP>/redfish/v1/TelemetryService/MetricDefinitions/<MetricID>
```

e.g. *<MetricID> = SystemMaxPowerConsumption*

```
{
    "@odata.type": "#MetricDefinition.v1_1_1.MetricDefinition",
    "@odata.context": "/redfish/v1/$metadata#MetricDefinition.MetricDefinition",
    "@odata.id":
    "/redfish/v1/TelemetryService/MetricDefinitions/SystemMaxPowerConsumption",
    "Id": "SystemMaxPowerConsumption",
    "Name": "System Max Power Consumption Metric Definition",
    "Description": "Peak system power consumption",
    "MetricType": "Numeric",
    "MetricDataType": "Decimal",
    "Units": "W",
    "Accuracy": 1,
    "SensingInterval": "PT60S",
}
```

B.1 AggregationMetrics Report

- SystemAvgInletTempHour
- SystemMaxInletTempHour
- SystemMaxPowerConsumption

B.2 CPUMemMetrics Report

- CPUC0ResidencyHigh
- CPUC0ResidencyLow
- CUPSIIOBandwidthDMI
- CUPSIIOBandwidthPort0
- CUPSIIOBandwidthPort1
- CUPSIIOBandwidthPort2
- CUPSIIOBandwidthPort3
- NonC0ResidencyHigh
- NonC0ResidencyLow

B.3 CPUSensor Report

- TemperatureReading

B.4 SystemUsage Report

- CPUUsage
- IOUsage
- MemoryUsage
- AggregateUsage

B.5 FanSensor Report

- RPMReading

B.6 FCPortStatistics Report

- FCInvalidCRCs
- FCLinkFailures
- FCLossOfSignals
- FCRxKBCount
- FCRxSequences
- FCRxTotalFrames
- FCTxKBCount
- FCTxSequences
- FCTxTotalFrames
- FCStatOSDriverState
- PortSpeed
- PortStatus

B.6 FCSensor Report

- TemperatureReading

B.7 FPGASensor Report

- TemperatureReading
- TotalFPGAPower

B.8 GPUMetrics Report

- BoardPowerSupplyStatus
- BoardTemperature
- GPUHealth
- GPUStatus
- MemoryTemperature
- PowerBrakeState

- PowerConsumption
- PowerSupplyStatus
- PrimaryTemperature
- SecondaryTemperature
- ThermalAlertState

B.9 GPUStatistics Report

- CumulativeDBECounterFB
- CumulativeDBECounterGR
- CumulativeSBECounterFB
- CumulativeSBECounterGR
- DBECounterFB
- DBECounterFBL2Cache
- DBECounterGRL1Cache
- DBECounterGRRF
- DBECounterGRTex
- DBERetiredPages
- SBECounterFB
- SBECounterFBL2Cache
- SBECounterGRL1Cache
- SBECounterGRRF
- SBECounterGRTex
- SBERetiredPages

B.10 MemorySensor Report

- TemperatureReading

B.11 NICSensor Report

- TemperatureReading

B.12 NICStatistics Report

- DiscardedPkts
- FCCRCErrorCount
- FCOELinkFailures
- FCOEPktRxCount
- FCOEPktTxCount
- FCOERxPktDroppedCount
- LanFCSRxErrors
- LanUnicastPktRxCount
- LanUnicastPktTxCount

- LinkStatus
- OSDriverState
- PartitionLinkStatus
- PartitionOSDriverState
- RDMARxTotalBytes
- RDMARxTotalPackets
- RDMATotalProtectionErrors
- RDMATotalProtocolErrors
- RDMATxTotalBytes
- RDMATxTotalPackets
- RDMATxTotalReadReqPkts
- RDMATxTotalSendPkts
- RDMATxTotalWritePkts
- RxBroadcast
- RxBytes
- RxErrorPktAlignmentErrors
- RxErrorPktFCSErrors
- RxFalseCarrierDetection
- RxJabberPkt
- RxMutlicast
- RxPauseXOFFFrames
- RxPauseXONFrames
- RxRuntPkt
- RxUnicast
- TxBroadcast
- TxBytes
- TxErrorPktExcessiveCollision
- TxErrorPktLateCollision
- TxErrorPktMultipleCollision
- TxErrorPktSingleCollision
- TxMutlicast
- TxPauseXOFFFrames
- TxPauseXONFrames
- TxUnicast

B.13 NVMeSMARTData Report

- AvailableSpare
- AvailableSpareThreshold
- CompositeTemparature
- ControllerBusyTimeLower
- ControllerBusyTimeUpper
- CriticalWarning

- DataUnitsReadLower
- DataUnitsReadUpper
- DataUnitsWrittenLower
- DataUnitsWrittenUpper
- HostReadCommandsLower
- HostReadCommandsUpper
- HostWriteCommandsLower
- HostWriteCommandsUpper
- MediaDataIntegrityErrorsLower
- MediaDataIntegrityErrorsUpper
- NumOfErrorInfoLogEntriesLower
- NumOfErrorInfoLogEntriesUpper
- PercentageUsed
- PowerCyclesLower
- PowerCyclesUpper
- PowerOnHoursLower
- PowerOnHoursUpper
- UnsafeShutdownsLower
- UnsafeShutdownsUpper

B.14 PowerMetrics Report

- SystemHeadRoomInstantaneous
- SystemInputPower
- SystemOutputPower
- SystemPowerConsumption
- TotalCPUPower
- TotalFanPower
- TotalMemoryPower
- TotalPciePower
- TotalStoragePower

B.15 PowerStatistics Report

- LastDayAvgPower
- LastDayMaxPower
- LastDayMaxPowerTime
- LastDayMinPower
- LastDayMinPowerTime
- LastHourAvgPower
- LastHourMaxPower
- LastHourMaxPowerTime
- LastHourMinPower

- LastHourMinPowerTime
- LastMinuteAvgPower
- LastMinuteMaxPower
- LastMinuteMaxPowerTime
- LastMinuteMinPower
- LastMinuteMinPowerTime
- LastWeekAvgPower
- LastWeekMaxPower
- LastWeekMaxPowerTime
- LastWeekMinPower
- LastWeekMinPowerTime

B.16 PSUMetrics Report

- FanSpeed
- Temperature

B.17 Sensor Report

- AmpsReading
- CPUUsagePctReading
- IOUsagePctReading
- MemoryUsagePctReading
- RPMReading
- SystemUsagePctReading
- TemperatureReading
- VoltageReading
- WattsReading

B.18 StorageDiskSMARTData Report

- CommandTimeout
- CRCErrorCount
- CurrentPendingSectorCount
- DriveTemperature
- ECCERate
- EraseFailCount
- ExceptionModeStatus
- MediaWriteCount
- PercentDriveLifeRemaining
- PowerCycleCount
- PowerOnHours
- ProgramFailCount
- ReadErrorRate

- ReallocatedBlockCount
- UncorrectableErrorCount
- UncorrectableLBACount
- UnusedReservedBlockCount
- UsedReservedBlockCount
- VolatileMemoryBackupSourceFailures

B.19 StorageSensor Report

- TemperatureReading

B.20 ThermalMetrics Report

- ComputePower
- ITUE
- PowerToCoolRatio
- PSUEfficiency
- SysAirFlowEfficiency
- SysAirflowPerFanPower
- SysAirflowPerSysInputPower
- SysAirflowUtilization
- SysNetAirflow
- SysRackTempDelta
- TotalPSUHeatDissipation

B.21 ThermalSensor Report

- TemperatureReading

C Sample Metric Report – *PowerMetrics*

```
{
        "@odata.type": "#MetricReport.v1_4_1.MetricReport",
        "@odata.context": "/redfish/v1/$metadata#MetricReport.MetricReport",
        "@odata.id": "/redfish/v1/TelemetryService/MetricReports/PowerMetrics",
        "Id": "PowerMetrics",
        "Name": "PowerMetrics Metric Report",
        "ReportSequence": "1",
        "Timestamp": "2021-05-26T20:00:56.000Z",
        "MetricReportDefinition": {
                "@odata.id":
"/redfish/v1/TelemetryService/MetricReportDefinitions/PowerMetrics"
        },
        "MetricValues": [
                {
                        "MetricId": "TotalMemoryPower",
                        "Timestamp": "2021-05-26T20:00:01.378Z",
                        "MetricValue": "1",
                        "Oem": {
                                "Dell": {
                                        "@odata.type":
"#DellMetricValue.v1_0_0.DellMetricValue",
                                        "ContextID": "PowerMetrics",
                                        "Label": "PowerMetrics TotalMemoryPower",
                                        "Source": "powermetrics",
                                        "FQDD": "PowerMetrics"
                                }
                        }
                },
                {
                        "MetricId": "TotalFanPower",
                        "Timestamp": "2021-05-26T20:00:01.378Z",
                        "MetricValue": "8.421875",
                        "Oem": {
                                "Dell": {
                                        "@odata.type":
"#DellMetricValue.v1_0_0.DellMetricValue",
                                        "ContextID": "PowerMetrics",
                                        "Label": "PowerMetrics TotalFanPower",
                                        "Source": "powermetrics",
                                        "FQDD": "PowerMetrics"
                                }
                        }
                },
                {
                        "MetricId": "SystemPowerConsumption",
                        "Timestamp": "2021-05-26T20:00:01.378Z",
                        "MetricValue": "126",
                        "Oem": {
                                "Dell": {
                                        "@odata.type":
"#DellMetricValue.v1_0_0.DellMetricValue",
                                        "ContextID": "PowerMetrics",
                                        "Label": "PowerMetrics
SystemPowerConsumption",
                                        "Source": "powermetrics",
                                        "FQDD": "PowerMetrics"
                                }
                        }
                },
                {
                    "MetricId": "SystemOutputPower",
                    "Timestamp": "2021-05-26T20:00:01.378Z",
                    "MetricValue": "112",
                    "Oem": {
                        "Dell": {
                            "@odata.type":
"#DellMetricValue.v1_0_0.DellMetricValue",
                            "ContextID": "PowerMetrics",
                            "Label": "PowerMetrics SystemOutputPower",
                            "Source": "powermetrics",
                            "FQDD": "PowerMetrics"
                        }
                    }
                },
                {
                    "MetricId": "TotalStoragePower",
                    "Timestamp": "2021-05-26T20:00:01.378Z",
                    "MetricValue": "12.890625",
                    "Oem": {
                        "Dell": {
                            "@odata.type":
"#DellMetricValue.v1_0_0.DellMetricValue",
                            "ContextID": "PowerMetrics",
                            "Label": "PowerMetrics TotalStoragePower",
                            "Source": "powermetrics",
                            "FQDD": "PowerMetrics"
                        }
                    }
                },
                {
                    "MetricId": "TotalPciePower",
                    "Timestamp": "2021-05-26T20:00:01.378Z",
                    "MetricValue": "0",
                    "Oem": {
                        "Dell": {
                            "@odata.type":
"#DellMetricValue.v1_0_0.DellMetricValue",
                            "ContextID": "PowerMetrics",
                            "Label": "PowerMetrics TotalPciePower",
                            "Source": "powermetrics",
                            "FQDD": "PowerMetrics"
                        }
                    }
                },
                {
                    "MetricId": "TotalFPGAPower",
                    "Timestamp": "2021-05-26T20:00:01.378Z",
                    "MetricValue": "0",
                                "Oem": {
                                        "Dell": {
                                                "@odata.type":
                "#DellMetricValue.v1_0_0.DellMetricValue",
                                                "ContextID": "PowerMetrics",
                                                "Label": "PowerMetrics TotalFPGAPower",
                                                "Source": "powermetrics",
                                                "FQDD": "PowerMetrics"
                                        }
                                }
                        },
                        {
                                "MetricId": "TotalCPUPower",
                                "Timestamp": "2021-05-26T20:00:01.378Z",
                                "MetricValue": "43",
                                "Oem": {
                                        "Dell": {
                                                "@odata.type":
                "#DellMetricValue.v1_0_0.DellMetricValue",
                                                "ContextID": "PowerMetrics",
                                                "Label": "PowerMetrics TotalCPUPower",
                                                "Source": "powermetrics",
                                                "FQDD": "PowerMetrics"
                                        }
                                }
                        },
                        {
                                "MetricId": "SystemHeadRoomInstantaneous",
                                "Timestamp": "2021-05-26T20:00:01.378Z",
                                "MetricValue": "624",
                                "Oem": {
                                        "Dell": {
                                                "@odata.type":
                "#DellMetricValue.v1_0_0.DellMetricValue",
                                                "ContextID": "PowerMetrics",
                                                "Label": "PowerMetrics
                SystemHeadRoomInstantaneous",
                                                "Source": "powermetrics",
                                                "FQDD": "PowerMetrics"
                                        }
                                }
                        },
                        {
                                "MetricId": "SystemInputPower",
                                "Timestamp": "2021-05-26T20:00:01.378Z",
                                "MetricValue": "126",
                                "Oem": {
                                        "Dell": {
```

```
                                        "@odata.type":
"#DellMetricValue.v1_0_0.DellMetricValue",
                                        "ContextID": "PowerMetrics",
                                        "Label": "PowerMetrics SystemInputPower",
                                        "Source": "powermetrics",
                                        "FQDD": "PowerMetrics"
                                }
                        }
                }
        ],
        "MetricValues@odata.count": 10
}
```