White Paper

Telemetry Streaming with iDRAC9—What you Need to Get Started

Abstract

Dell EMC PowerEdge Servers with iDRAC9 4.0 Datacenter streams 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, and provides troubleshooting tips.

March 2020

Revisions

Date	Description
November 2019	Initial release
June 2020	Errata update

Acknowledgments

Authors: Sankara Gara, Cyril Jose, Sailaja Mahendrakar, Praveen Thangavelu, MaheshBabu Ramaiah, Doug Iler

Table of contents

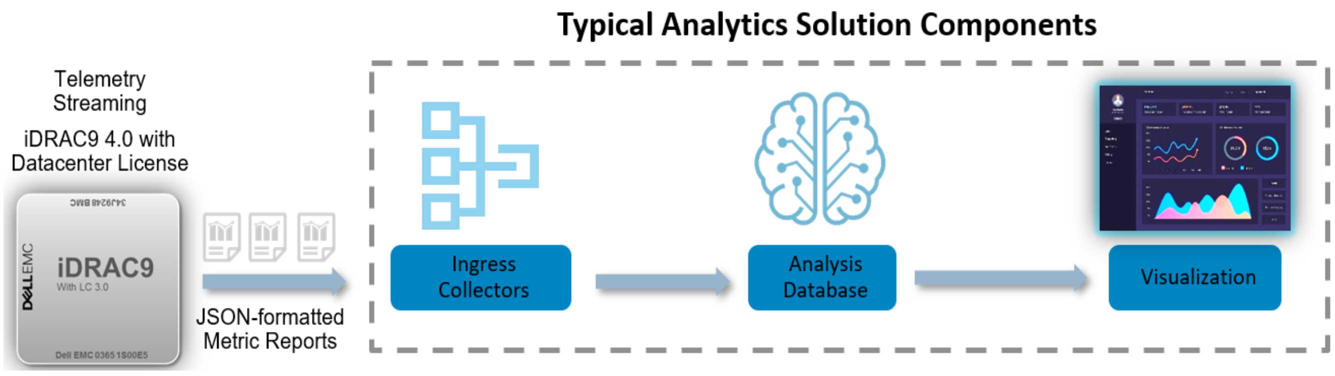
Executive summary

With iDRAC9 v4.00.00.00 firmware and a 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.

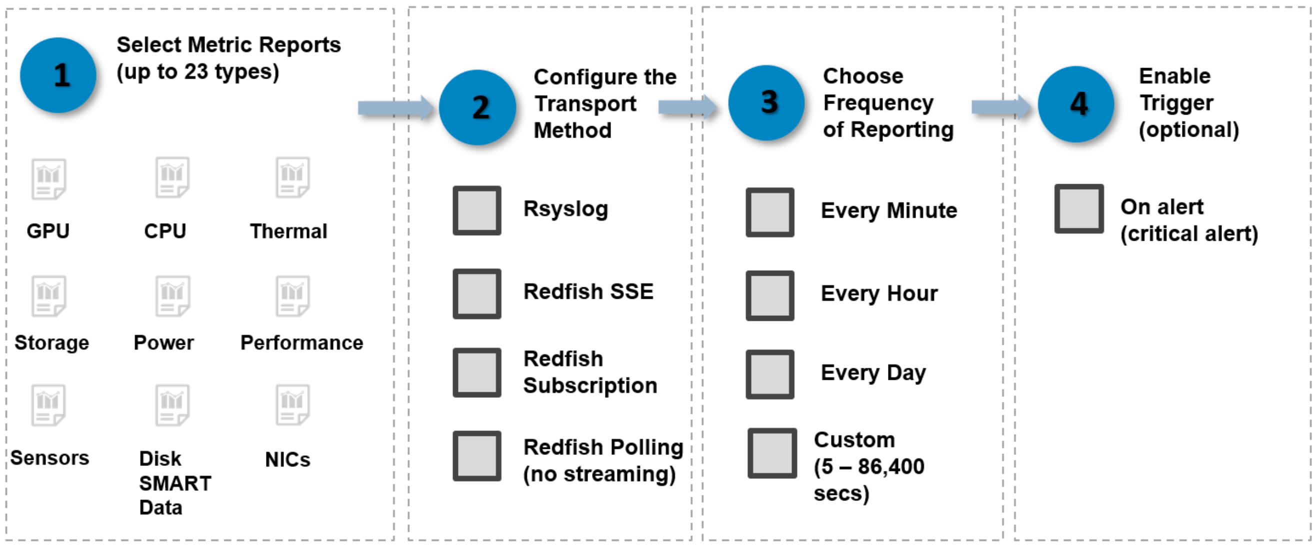
1 Telemetry overview

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 one or more PowerEdge servers 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
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 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.

Remote syslog (RSyslog): Remote syslog implements the basic syslog protocol, and extends it with content-based filtering, rich filtering capabilities, and flexible configuration options.

EEMI: The *Event and Error Message Information* is 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.

1.2 Prerequisites

The Telemetry feature is available on iDRAC9 firmware version 4.00.00.00 or above and requires a Datacenter license.

2 Configuring telemetry

Telemetry configuration allows you to configure telemetry data streaming behavior and report generation behavior. It includes the global settings common to all reports, and those settings specific to each available report. Enabling or disabling telemetry at the global setting level enables or disables all reports for telemetry streaming. By default, the telemetry feature is disabled at the global setting level and for all reports individually. A simple configuration includes enabling telemetry at the global setting, each wanted report, and the report interval for each report.

By default, telemetry reports are sent to connected Redfish clients using the HTTP protocol. To receive reports using the syslog protocol, configure the Remote Syslog (RSyslog) settings. Typically, reports are streamed on a configured *ReportInterval* condition, but they can also be streamed at error or warning conditions. Trigger definitions are based on iDRAC life-cycle events that are generated for error and warning conditions.

Table 1 Global settings

Setting	**Description**
EnableTelemetry	Enable or disable telemetry globally
RsyslogServer1	Remote syslog server 1 address IPv4, IPV6 or FQDN
RsyslogServer1Port	Remote syslog server 1 port
RsyslogServer2	Remote syslog server 2 address IPv4, IPV6, or FQDN
RsyslogServer2Port	Remote syslog server 2 port
TelemetrySubscription1	Redfish subscription (SSE or Post to Subscription)*
TelemetrySubscription2	Redfish subscription (SSE or Post to Subscription)*

Table 2 Per report settings

Setting	**Description**
EnableTelemetry	Enable or disable telemetry for a report
ReportInterval	Specify when reports are pushed
ReportTriggers	Specify triggers for generating a report
RsyslogTarget	Specify if the report should be sent to a Rsyslog server also
DevicePollFrequency	How often the device or source is polled for data. *

* read-only

Table 3 Supported reports

AggregationMetrics
CUPS
GPUMetrics
NICStatistics
PSUMetrics
ThermalMetrics
CPUMemMetrics
FanSensor
GPUStatistics
NVMeSMARTData
Sensor
ThermalSensor
CPURegisters
FCSensor
MemorySensor

PowerMetrics
StorageDiskSMARTData
CPUSensor
FPGASensor
NICSensor
PowerStatistics
StorageSensor

Table 4 Supported triggers

CPUCriticalTrigger
MEMWarnTrigger
TMPCpuWarnTrigger
CPUWarnTrigger
NVMeCriticalTrigger
TMPCriticalTrigger
FANCriticalTrigger
NVMeWarnTrigger
TMPDiskCriticalTrigger
FANWarnTrigger
PDRCriticalTrigger
MPDiskWarnTrigger
IERRCriticalTrigger
PDRWarnTrigger
TMPWarnTrigger
MEMCriticalTrigger
TMPCpuCriticalTrigger
VLTCriticalTrigger

2.1 Workflow example configuring telemetry using Redfish

Global:

```
HTTP PATCH /redfish/v1/Managers/iDRAC.Embedded.1/Attributes
Payload: {"Attributes":{"Telemetry.1.<attribute>": "<value>"}

e.g.

curl -s -k -u user:pw -X PATCH https://<IDRAC-
IP>/redfish/v1/Managers/iDRAC.Embedded.1/Attributes
-H 'Content-Type: application/json' -d '{ "Attributes":
{"Telemetry.1.EnableTelemetry": "Enabled"}}'
```

Per report:

```
HTTP PATCH /redfish/v1/Managers/iDRAC.Embedded.1/Attributes
Payload: {"Attributes":{"Telemetry<report>.1.<attribute>": "<value>"}

e.g.

curl -s -k -u user:pw -X PATCH https://<IDRAC-
IP>/redfish/v1/Managers/iDRAC.Embedded.1/Attributes
```

```
-H 'Content-Type: application/json' -d '{"Attributes":
{"TelemetryPowerMetrics.1.EnableTelemetry":"Enabled"}}'
```

2.2 Workflow example configuring telemetry using RACADM

The global and per report configuration can be done using iDRAC RACADM get and set operations on the Telemetry configuration attributes. The following is an example of enabling Telemetry on global and per report (*PowerMetrics*) and setting report interval in seconds on the report.

Global:

```
racadm get idrac.telemetry
racadm set idrac.telemetry.<attribute> <value>
```

e.g.

```
racadm set idrac.telemetry.enabletelemetry Enabled
```

Per report:

```
racadm get idrac.telemetry<report>.1
racadm set idrac.telemetry<report>.1.<attribute> <value>
```

e.g.

```
racadm set idrac.telemetrypowermetrics.1.enabletelemetry Enabled
racadm set idrac.telemetrypowermetrics.1.reportinterval 300
```

2.2.1 Workflow example configuring telemetry using SCP

Export of “IDRAC” component will include the Telemetry attributes (as shown in the configuration XML snippet below). The desired settings are updated in the export XML file and imported.

```
<Attribute Name="Telemetry.1#EnableTelemetry">Enabled</Attribute>
<Attribute Name="Telemetry.1#RSyslogServer1">10.35.xxx.xxx</Attribute>
<Attribute Name="Telemetry.1#RSyslogServer1Port">xxxx</Attribute>
<Attribute Name="Telemetry.1#RSyslogServer2">10.35.xxx.xxx</Attribute>
<Attribute Name="Telemetry.1#RSyslogServer2Port">xxxx</Attribute>
<Attribute Name="TelemetryCPUSensor.1#EnableTelemetry">Enabled</Attribute>
<Attribute Name="TelemetryCPUSensor.1#ReportInterval">600</Attribute>
<Attribute Name="TelemetryCPUSensor.1#RsyslogTarget">TRUE</Attribute>
<Attribute Name="TelemetryCPUSensor.1#ReportTriggers">
              TMPCpuCriticalTrigger,TMPCpuWarnTrigge</Attribute>
```

2.2.2 Workflow example configuring RSyslog

Redfish:

```
HTTP PATCH   /redfish/v1/Managers/iDRAC.Embedded.1/Attributes
Payload: {"Attributes":{"Telemetry.1.RsyslogServer1": "<ip/fqdn>",
"Telemetry.1.RsyslogServer1port": "<port>"}
```

```
HTTP POST https://<idrac-
ip>/redfish/v1/Dell/Managers/iDRAC.Embedded.1/DelliDRACCardService/Actions/Delli
DRACCardService.TestRsyslogServerConnection

HTTP PATCH /redfish/v1/Managers/iDRAC.Embedded.1/Attributes
Payload: {"Attributes":{"Telemetry<report>.1.RsyslogTarget": "True"}
```

e.g.

```
curl -s -k -u user:pw -X PATCH https://<IDRAC-
IP>/redfish/v1/Managers/iDRAC.Embedded.1/Attributes
-H 'Content-Type: application/json' -d
'{"Attributes":{"Telemetry.1.RsyslogServer1": "1.1.1.1",
"Telemetry.1.RsyslogServer1port": "10514"}'

curl -s -k -u user:pw -X PATCH https://<IDRAC-
IP>/redfish/v1/Managers/iDRAC.Embedded.1/DelliDRACCardService/Actions/DelliDRACC
ardService.TestRsyslogServerConnection

curl -s -k -u user:pw -X PATCH https://<IDRAC-
IP>/redfish/v1/Managers/iDRAC.Embedded.1/Attributes
-H 'Content-Type: application/json' -d '{ "Attributes":
{"TelemetryPowerMetrics.1.RsyslogTarget": "True"}}'
```

RACADM:

```
racadm set idrac.telemetry.RsyslogServer1 "<ip/fqdn>"
racadm set idrac.telemetry.RsyslogServer1port "<port>"
racadm testrsyslogconnection

racadm set idrac.telemetry<report>.1.RsyslogTarget True
e.g.
racadm set idrac.telemetryPowerMetrics.1.RsyslogTarget True
racadm set idrac.telemetry.RsyslogServer1 "1.1.1.1"
racadm set idrac.telemetry.RsyslogServer1port "10514"
racadm testrsyslogconnection
racadm set idrac.telemetryPowerMetrics.1.RsyslogTarget True
```

2.2.3 Workflow example configuring triggers

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.

Redfish:

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

e.g.
curl -s -k -u user:pw -X PATCH https://<IDRAC-
IP>/redfish/v1/Managers/iDRAC.Embedded.1/Attributes
```

```
-H 'Content-Type: application/json' -d '{ "Attributes":
{"TelemetryPowerMetrics.1.ReportTriggers": "CPUCriticalTrigger,
CPUWarnTrigger"}}'
```

RACADM:

```
racadm set idrac.telemetry<report>.1.ReportTriggers "<tri1, trig2>"
```

e.g.

```
racadm set idrac.telemetryPowerMetrics.1.ReportTriggers "CPUCriticalTrigger,
CPUWarnTrigger"
```

3 Receiving telemetry reports

After telemetry streaming is configured on the iDRAC, telemetry reports are streamed to the configured Redfish clients or Remote Syslog servers. The Redfish client can also pull the reports on demand. The following sections describe the methods through which clients can receive the data.

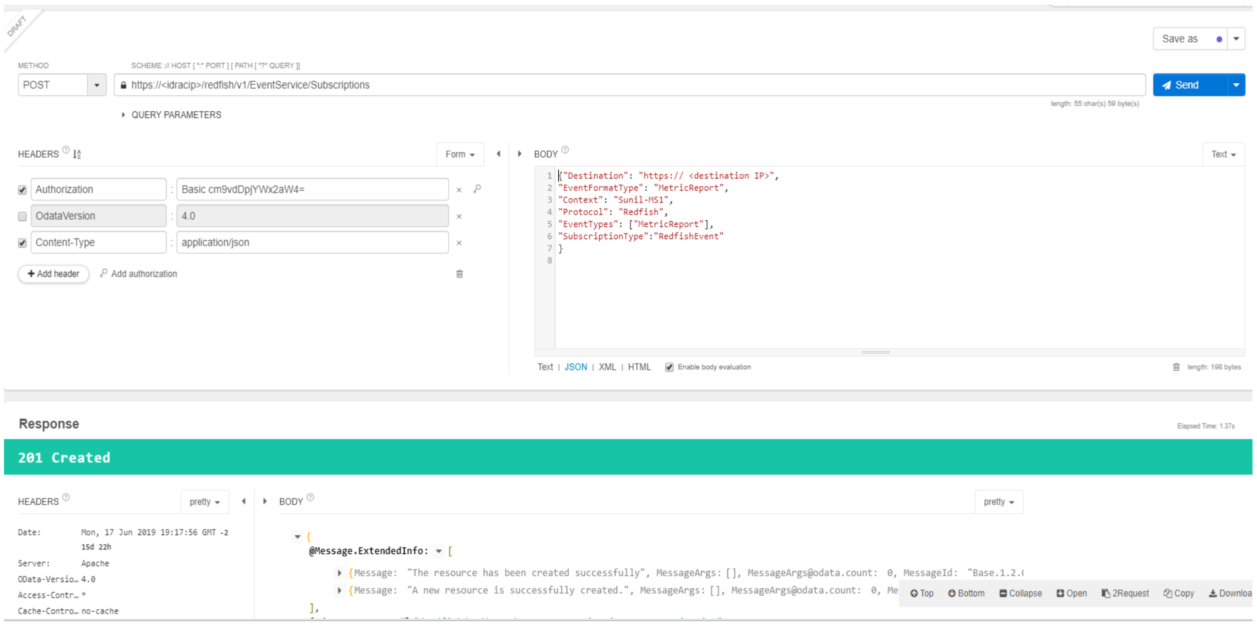
3.1 Redfish client using subscription method

A Redfish client receives the telemetry reports using subscription method. The first step is to create a subscription specifying the destination using HTTP POST request. At the report interval or trigger condition, if triggers are configured, all enabled reports are sent to the destination specified in the subscription request. The HTTP client can receive reports by listening on a destination port.

```
POST https://<IDRAC-IP>/redfish/v1/EventService/Subscriptions

Body: {

"Destination": "https://<listener ip:port>",

"EventFormatType": "MetricReport",

"Context": "TelmetryTest",

"Protocol": "Redfish",

"EventTypes": ["MetricReport"],

"SubscriptionType":"RedfishEvent"

}
```

Clients can terminate subscriptions by sending an HTTP DELETE message to the iDRAC Event Service. The number of subscriptions a user can create is a maximum of 2.

The following figures show the HTTP requests for creating, deleting and getting subscriptions.



METHOD

POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

https://<idracip>/redfish/v1/EventService/Subscriptions

length: 55 char(s) 59 byte(s)

HEADERS

✓	Authorization	Basic cm9vdDpjYWx2aW4=
	OdataVersion	4.0
✓	Content-Type	application/json

BODY

```
{"Destination": "https:// <destination IP>",
"EventFormatType": "MetricReport",
"Context": "Sunil-MS1",
"Protocol": "Redfish",
"EventTypes": ["MetricReport"],
"SubscriptionType":"RedfishEvent"
}
```

Text | JSON | XML | HTML ✓ Enable body evaluation length: 198 bytes

Response

Elapsed Time: 1.37s

201 Created

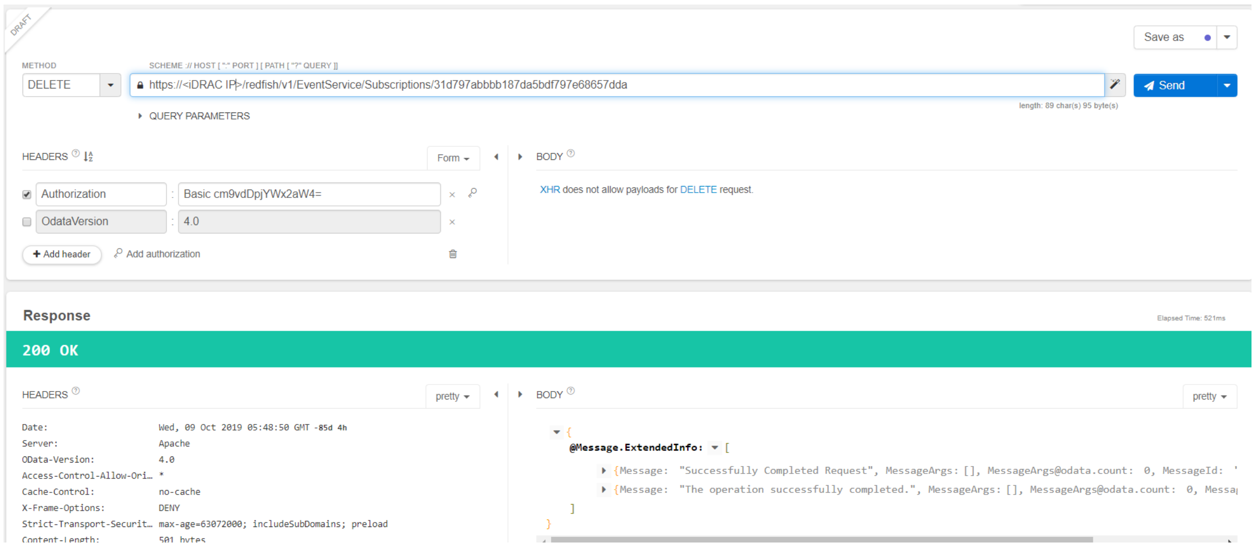
HEADERS

```
Date:          Mon, 17 Jun 2019 19:17:56 GMT -2
               15d 22h
Server:        Apache
OData-Versio…  4.0
Access-Contr…  *
Cache-Contro…  no-cache
```

BODY

```
{
  @Message.ExtendedInfo: [
    {Message:  "The resource has been created successfully", MessageArgs: [], MessageArgs@odata.count:  0, MessageId:  "Base.1.2.(
    {Message:  "A new resource is successfully created.", MessageArgs: [], MessageArgs@odata.count:  0, Me
  ],
```

Figure 1 Create subscription request



METHOD

DELETE

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

https://<iDRAC IP>/redfish/v1/EventService/Subscriptions/31d797abbbb187da5bdf797e68657dda

length: 89 char(s) 95 byte(s)

HEADERS

✓	Authorization	Basic cm9vdDpjYWx2aW4=
	OdataVersion	4.0

BODY

XHR does not allow payloads for DELETE request.

Response

Elapsed Time: 521ms

200 OK

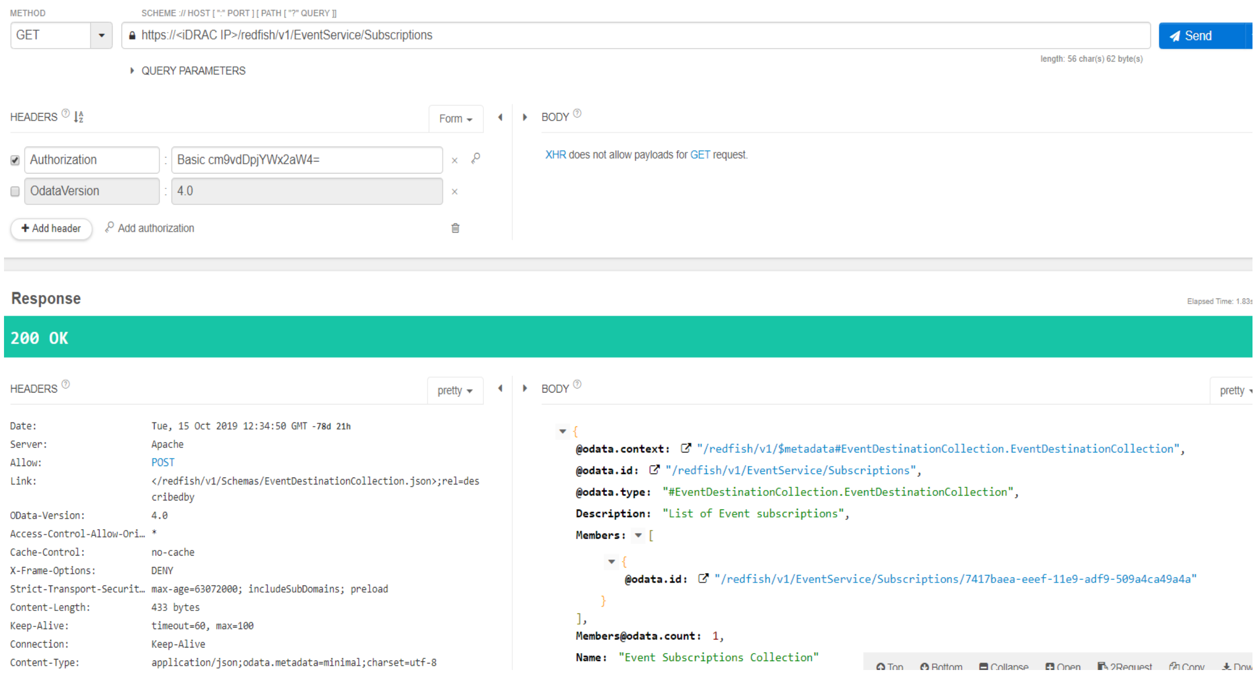
HEADERS

```
Date:                        Wed, 09 Oct 2019 05:48:50 GMT -85d 4h
Server:                      Apache
OData-Version:               4.0
Access-Control-Allow-Ori…    *
Cache-Control:               no-cache
X-Frame-Options:             DENY
Strict-Transport-Securit…    max-age=63072000; includeSubDomains; preload
Content-Length:              501 bytes
```

BODY

```
{
  @Message.ExtendedInfo: [
    {Message:  "Successfully Completed Request", MessageArgs: [], MessageArgs@odata.count:  0, MessageId:  '
    {Message:  "The operation successfully completed.", MessageArgs: [], MessageArgs@odata.count:  0, Messa
  ]
}
```

Figure 2 Delete subscription request


METHOD GET · SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

https://<iDRAC IP>/redfish/v1/EventService/Subscriptions
Send

length: 56 char(s) 62 byte(s)

QUERY PARAMETERS

HEADERS · Form

✓	Authorization	Basic cm9vdDpjYWx2aW4=
	OdataVersion	4.0

Add header · Add authorization

BODY: XHR does not allow payloads for GET request.

Response · Elapsed Time: 1.83s

200 OK

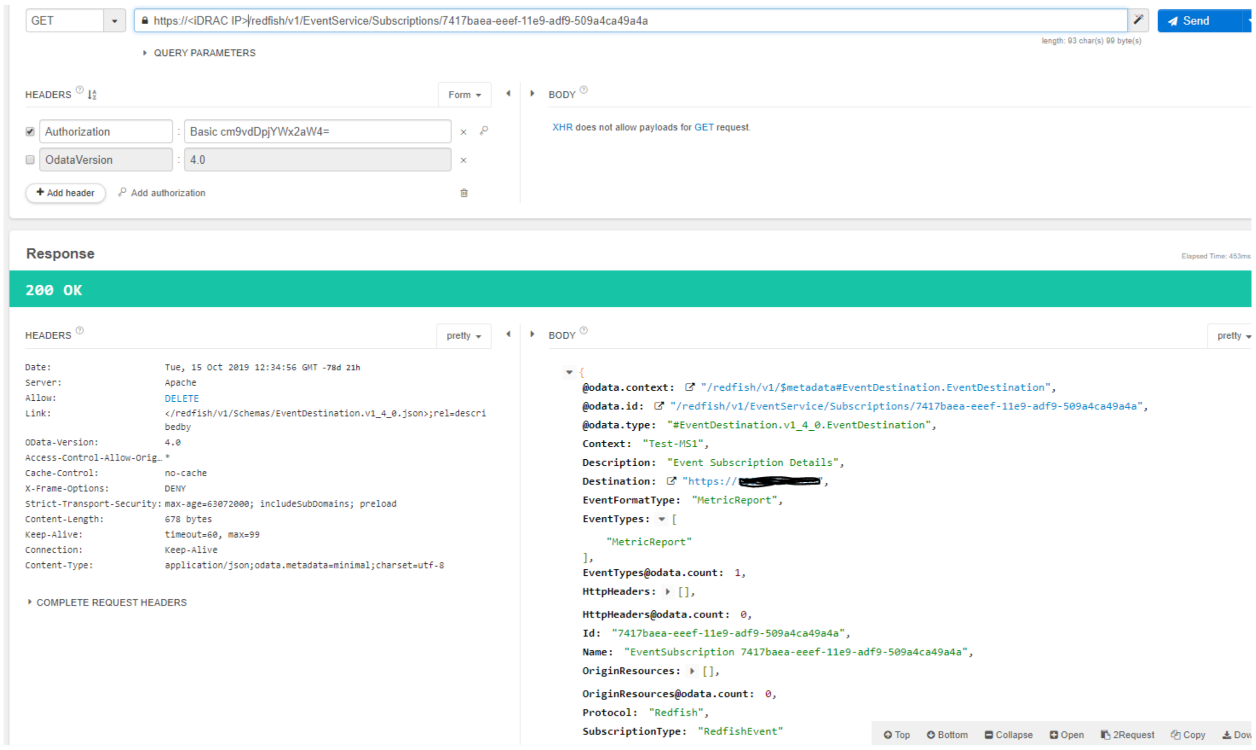
HEADERS

```
Date:                         Tue, 15 Oct 2019 12:34:50 GMT -78d 21h
Server:                       Apache
Allow:                        POST
Link:                         </redfish/v1/Schemas/EventDestinationCollection.json>;rel=des
                              cribedby
OData-Version:                4.0
Access-Control-Allow-Ori…     *
Cache-Control:                no-cache
X-Frame-Options:              DENY
Strict-Transport-Securit…     max-age=63072000; includeSubDomains; preload
Content-Length:               433 bytes
Keep-Alive:                   timeout=60, max=100
Connection:                   Keep-Alive
Content-Type:                 application/json;odata.metadata=minimal;charset=utf-8
```

BODY

```
{
  @odata.context: "/redfish/v1/$metadata#EventDestinationCollection.EventDestinationCollection",
  @odata.id: "/redfish/v1/EventService/Subscriptions",
  @odata.type: "#EventDestinationCollection.EventDestinationCollection",
  Description: "List of Event subscriptions",
  Members: [
      {
        @odata.id: "/redfish/v1/EventService/Subscriptions/7417baea-eeef-11e9-adf9-509a4ca49a4a"
      }
  ],
  Members@odata.count: 1,
  Name: "Event Subscriptions Collection"
```

Figure 3 Getting subscription collection


GET https://<iDRAC IP>/redfish/v1/EventService/Subscriptions/7417baea-eeef-11e9-adf9-509a4ca49a4a
Send

length: 93 char(s) 99 byte(s)

QUERY PARAMETERS

HEADERS · Form

✓	Authorization	Basic cm9vdDpjYWx2aW4=
	OdataVersion	4.0

Add header · Add authorization

BODY: XHR does not allow payloads for GET request.

Response · Elapsed Time: 453ms

200 OK

HEADERS

```
Date:                         Tue, 15 Oct 2019 12:34:56 GMT -78d 21h
Server:                       Apache
Allow:                        DELETE
Link:                         </redfish/v1/Schemas/EventDestination.v1_4_0.json>;rel=descri
                              bedby
OData-Version:                4.0
Access-Control-Allow-Orig…    *
Cache-Control:                no-cache
X-Frame-Options:              DENY
Strict-Transport-Security:    max-age=63072000; includeSubDomains; preload
Content-Length:               678 bytes
Keep-Alive:                   timeout=60, max=99
Connection:                   Keep-Alive
Content-Type:                 application/json;odata.metadata=minimal;charset=utf-8
```

COMPLETE REQUEST HEADERS

BODY

```
{
  @odata.context: "/redfish/v1/$metadata#EventDestination.EventDestination",
  @odata.id: "/redfish/v1/EventService/Subscriptions/7417baea-eeef-11e9-adf9-509a4ca49a4a",
  @odata.type: "#EventDestination.v1_4_0.EventDestination",
  Context: "Test-MS1",
  Description: "Event Subscription Details",
  Destination: "https://[redacted]",
  EventFormatType: "MetricReport",
  EventTypes: [
      "MetricReport"
  ],
  EventTypes@odata.count: 1,
  HttpHeaders: [],
  HttpHeaders@odata.count: 0,
  Id: "7417baea-eeef-11e9-adf9-509a4ca49a4a",
  Name: "EventSubscription 7417baea-eeef-11e9-adf9-509a4ca49a4a",
  OriginResources: [],
  OriginResources@odata.count: 0,
  Protocol: "Redfish",
  SubscriptionType: "RedfishEvent"
```

Figure 4 Getting subscription details:

3.1.1 Redfish client using SSE method

The SSE method is one more way of streaming telemetry data, with Redfish client and iDRAC event service communicating using the HTML5 SSE feature and HTTP protocol. The client receives Telemetry data on performing a GET on SSE URI. The streaming URI contains the event format type as metric report, which directs the iDRAC event service to stream enabled metric reports alone. The client-triggered SSE URI can also be provisioned to query specific metric reports that are streamed with the use of $filter. Upon receipt of a GET request from the client, the Event service makes an entry of the SSE request in the subscription collection and starts streaming the reports that are enabled and configured for triggers. The message received on the client contains the fields defined by the SSE protocol, namely Event, Id, Data and Retry. The metric report content is bundled in the Datasection and the Id contains the report sequence for the streamed content. The Timestamp behavior of metric reports documented in the subscription method holds good for SSE as well.

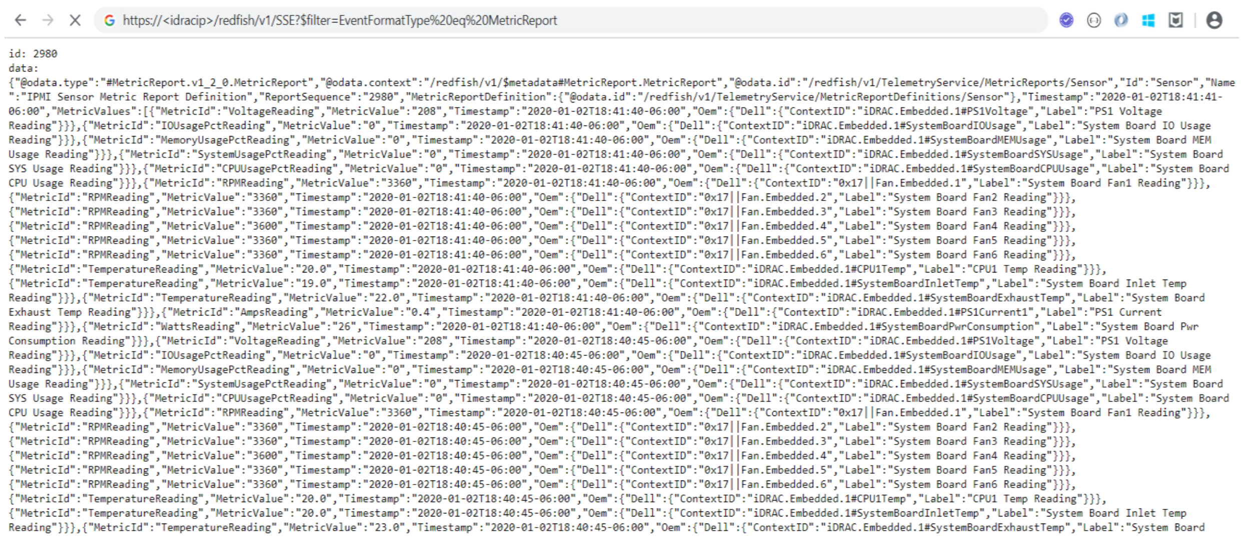
The connection can be terminated by either the client or iDRAC event service. Where there is no telemetry data sent to client for more than an hour, the connection is closed from the event service endpoint. Where connections drop off due to a network glitch or for unknown reasons, then the last event id is provided by the client to the event service to resume streaming. When the connection closes, the subscription entry is removed from the list. A maximum of 2 SSE connections are allowed at one time, and any request beyond this is not honored.

HTTP request to stream all reports:

```
HTTP GET /redfish/v1/SSE?$filter=EventFormatType%20eq%20MetricReport
e.g.
curl -s -k -u user:pw -X GET https://<IDRAC-
IP>/redfish/v1/SSE?$filter=EventFormatType%20eq%20MetricReport
```

HTTP request to stream a single report:

```
HTTP GET
/redfish/v1/SSE?$filter=MetricReportDefinition%20eq%20'/redfish/v1/TelemetryService/Metri
cReportDefinitions/<report>'


e.g.
curl -s -k -u user:pw -X GET
https://<IDRAC-
IP>/redfish/v1/SSE?$filter=MetricReportDefinition%20eq%20'/redfish/v1/TelemetryService/Me
tricReportDefinitions/PowerMetrics'
```



https://<idracip>/redfish/v1/SSE?$filter=EventFormatType%20eq%20MetricReport

```
id: 2980
data:
{"@odata.type":"#MetricReport.v1_2_0.MetricReport","@odata.context":"/redfish/v1/$metadata#MetricReport.MetricReport","@odata.id":"/redfish/v1/TelemetryService/MetricReports/Sensor","Id":"Sensor","Name
":"IPMI Sensor Metric Report Definition","ReportSequence":"2980","MetricReportDefinition":{"@odata.id":"/redfish/v1/TelemetryService/MetricReportDefinitions/Sensor"},"Timestamp":"2020-01-02T18:41:41-
06:00","MetricValues":[{"MetricId":"VoltageReading","MetricValue":"208","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#PS1Voltage","Label":"PS1 Voltage
Reading"}}},{"MetricId":"IOUsagePctReading","MetricValue":"0","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardIOUsage","Label":"System Board IO Usage
Reading"}}},{"MetricId":"MemoryUsagePctReading","MetricValue":"0","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardMEMUsage","Label":"System Board MEM
Usage Reading"}}},{"MetricId":"SystemUsagePctReading","MetricValue":"0","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardSYSUsage","Label":"System Board
SYS Usage Reading"}}},{"MetricId":"CPUUsagePctReading","MetricValue":"0","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardCPUUsage","Label":"System Board
CPU Usage Reading"}}},{"MetricId":"RPMReading","MetricValue":"3360","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.1","Label":"System Board Fan1 Reading"}}},
{"MetricId":"RPMReading","MetricValue":"3360","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.2","Label":"System Board Fan2 Reading"}}},
{"MetricId":"RPMReading","MetricValue":"3360","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.3","Label":"System Board Fan3 Reading"}}},
{"MetricId":"RPMReading","MetricValue":"3600","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.4","Label":"System Board Fan4 Reading"}}},
{"MetricId":"RPMReading","MetricValue":"3360","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.5","Label":"System Board Fan5 Reading"}}},
{"MetricId":"RPMReading","MetricValue":"3360","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.6","Label":"System Board Fan6 Reading"}}},
{"MetricId":"TemperatureReading","MetricValue":"20.0","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#CPU1Temp","Label":"CPU1 Temp Reading"}}},
{"MetricId":"TemperatureReading","MetricValue":"19.0","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardInletTemp","Label":"System Board Inlet Temp
Reading"}}},{"MetricId":"TemperatureReading","MetricValue":"22.0","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardExhaustTemp","Label":"System Board
Exhaust Temp Reading"}}},{"MetricId":"AmpsReading","MetricValue":"0.4","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#PS1Current1","Label":"PS1 Current
Reading"}}},{"MetricId":"WattsReading","MetricValue":"26","Timestamp":"2020-01-02T18:41:40-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardPwrConsumption","Label":"System Board Pwr
Consumption Reading"}}},{"MetricId":"VoltageReading","MetricValue":"208","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#PS1Voltage","Label":"PS1 Voltage
Reading"}}},{"MetricId":"IOUsagePctReading","MetricValue":"0","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardIOUsage","Label":"System Board IO Usage
Reading"}}},{"MetricId":"MemoryUsagePctReading","MetricValue":"0","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardMEMUsage","Label":"System Board MEM
Usage Reading"}}},{"MetricId":"SystemUsagePctReading","MetricValue":"0","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardSYSUsage","Label":"System Board
SYS Usage Reading"}}},{"MetricId":"CPUUsagePctReading","MetricValue":"0","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardCPUUsage","Label":"System Board
CPU Usage Reading"}}},{"MetricId":"RPMReading","MetricValue":"3360","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.1","Label":"System Board Fan1 Reading"}}},
{"MetricId":"RPMReading","MetricValue":"3360","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.2","Label":"System Board Fan2 Reading"}}},
{"MetricId":"RPMReading","MetricValue":"3360","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.3","Label":"System Board Fan3 Reading"}}},
{"MetricId":"RPMReading","MetricValue":"3600","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.4","Label":"System Board Fan4 Reading"}}},
{"MetricId":"RPMReading","MetricValue":"3360","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.5","Label":"System Board Fan5 Reading"}}},
{"MetricId":"RPMReading","MetricValue":"3360","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"0x17||Fan.Embedded.6","Label":"System Board Fan6 Reading"}}},
{"MetricId":"TemperatureReading","MetricValue":"20.0","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#CPU1Temp","Label":"CPU1 Temp Reading"}}},
{"MetricId":"TemperatureReading","MetricValue":"20.0","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardInletTemp","Label":"System Board Inlet Temp
Reading"}}},{"MetricId":"TemperatureReading","MetricValue":"23.0","Timestamp":"2020-01-02T18:40:45-06:00","Oem":{"Dell":{"ContextID":"iDRAC.Embedded.1#SystemBoardExhaustTemp","Label":"System Board
```

Figure 5 Redfish client receiving reports SSE method

3.1.2 Redfish Client Using 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 One Report:

```
HTTP GET /redfish/v1/TelemetryService/MetricReports/<report>
e.g.
curl -s -k -u user:pw -X GET https://<IDRAC-
IP>/redfish/v1/TelemetryService/MetricReports/PowerMetrics
```

Pull Report Collection (URI list only):

```
HTTP GET /redfish/v1/TelemetryService/MetricReports
e.g.
curl -s -k -u user:pw -X GET https://<IDRAC-
IP>/redfish/v1/TelemetryService/MetricReports
```

Previously generated reports are returned in response to the above GET command at the report interval or when a trigger condition is met.

3.1.3 Remote syslog server using syslog protocol

After the Rsyslog server, port, and reports are configured, the remote syslog server will receive the telemetry report in the body of the syslog message. Large reports are received in multiple syslog messages. These reports contain information such as sequence numbers, totals chunks, and chunk number; all of which are used to reconstruct the whole report. For details on the server-side configuration, see the white paper, *Telemetry Streaming MetricReport Using Rsyslog*. The paper also includes information about configuring SSL certificates and how to process the report information in the syslog message.

3.1.4 Report generation behavior and limitations

General report behavior: The number of data instances cached and used for a report generation depends on the *DevicePollFrequency* and *ReportInterval* configured on a report. For example, if *DevicePollFrequency* is 5 seconds and *ReportInterval* is 60 seconds, 12 data instances are used to create a report. The report will be part of report collection, and available for the PULL request until another report replaces it on next *ReportInterval*. The above sample count is for a single instance per device poll. In multiple instances of data per device poll, the instance, or sample count, is 12 times the runtime devices instances.

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 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*.

For all metric reports, the users can set a *ReportInterval* of 0 apart from the defined boundary values. When the *ReportInterval* is set to 0, the report can only be pulled and it cannot be streamed.

3.2 Troubleshooting and Tips

Problem	Possible Solutions
1. Cannot configure telemetry.	1. Check license. Need "Datacenter" license for most reports. 2. "OpenManage Enterprise" license for basic reports.
2. Redfish client cannot stream or pull telemetry reports.	1. Check license. 2. Check if global Telemetry enabled. 3. Check per report Telemetry enabled. 4. Check User Guide for unsupported ports for Subscription destination. 5. Check if host is powered off.
3. No reports are seen Rsyslog server.	1. Check if Rsyslog target set to true on the report settings. 2. Check if Rsyslog connection is good. 3. Check if host is powered off.
4. No reports are streamed on Trigger.	1. Check Redfish eventing setting in IDRAC GUI. 2. Check LC log for the EEMI ID that belongs to trigger. 3. Check User Guide for triggers to EEMI ID mapping. 4. Check if host is powered off.

3.3 Best practices

1. A Server Configuration Profile (SCP) would be a better option to configure all the metric reports, setting *ReportInterval* and enabling *RSyslogTarget*. 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 can in-turn result in large telemetry reports since it includes every relevant device metric. Also, a minimum report interval like 5 s 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 advisable to NOT set the ReportInterval to maximum value.

b. For reports like *CUPS, PowerMetrics, CPUMemMetrics, ThermalMetrics, and GPUMetrics*, it is recommended to set a minimum *ReportInterval* of 60 s even though the minimum *ReportInterval* of 5 s is allowed.

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

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 CUPS Report

- CPUUsage
- IOUsage
- MemoryUsage
- SystemUsage

B.5 FanSensor Report

- RPMReading

B.6 FCSensor Report

- TemperatureReading

B.7 FPGASensor Report

- TemperatureReading

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

MetricIDs

- SysAirflowUtilization
- SysNetAirflow
- SysRackTempDelta
- TotalPSUHeatDissipation

B.21 ThermalSensor Report

- TemperatureReading

C Sample Metric Report - *PowerMetrics*

```
{
    "@odata.type": "#MetricReport.v1_2_0.MetricReport",
    "@odata.context": "/redfish/v1/$metadata#MetricReport.MetricReport,"
    "@odata.id": "/redfish/v1/TelemetryService/MetricReports/PowerMetrics",
    "Id": "PowerMetrics",
    "Name": "Power Metrics Metric Report,"
    "ReportSequence": "1",
    "MetricReportDefinition": {
        "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/PowerMetrics"
    },
    "Timestamp": "2020-02-03T20:10:58-06:00",
    "MetricValues": [
        {
            "MetricId": "SystemHeadRoomInstantaneous",
            "Timestamp": "2020-02-03T20:10:24-06:00",
            "MetricValue": "642",
            "Oem": {
                "Dell": {
                    "ContextID": "PowerMetrics",
                    "Label": "PowerMetrics SystemHeadRoomInstantaneous"
                }
            }
        },
        {
            "MetricId": "SystemInputPower",
            "Timestamp": "2020-02-03T20:10:24-06:00",
            "MetricValue": "108",
            "Oem": {
                "Dell": {
                    "ContextID": "PowerMetrics",
                    "Label": "PowerMetrics SystemInputPower"
                }
            }
        },
        {
            "MetricId": "SystemOutputPower",
            "Timestamp": "2020-02-03T20:10:24-06:00",
            "MetricValue": "94",
            "Oem": {
                "Dell": {
                    "ContextID": "PowerMetrics",
                    "Label": "PowerMetrics SystemOutputPower"
                }
            }
        },
        {
            "MetricId": "SystemPowerConsumption",
            "Timestamp": "2020-02-03T20:10:24-06:00",
            "MetricValue": "108",
            "Oem": {
                "Dell": {
                    "ContextID": "PowerMetrics",
                    "Label": "PowerMetrics SystemPowerConsumption"
                }
            }
        },
        {
            "MetricId": "TotalCPUPower",
            "Timestamp": "2020-02-03T20:10:24-06:00",
            "MetricValue": "58.0",
            "Oem": {
                "Dell": {
                    "ContextID": "PowerMetrics",
                    "Label": "PowerMetrics TotalCPUPower"
                }
            }
        },
        {
            "MetricId": "TotalFanPower",
            "Timestamp": "2020-02-03T20:10:24-06:00",
            "MetricValue": "4.890625",
            "Oem": {
                "Dell": {
                    "ContextID": "PowerMetrics",
                    "Label": "PowerMetrics TotalFanPower"
                }
            }
        },
        {
            "MetricId": "TotalMemoryPower",
            "Timestamp": "2020-02-03T20:10:24-06:00",
            "MetricValue": "2.0",
            "Oem": {
                "Dell": {
                    "ContextID": "PowerMetrics",
                    "Label": "PowerMetrics TotalMemoryPower"
                }
            }
        },
        {
            "MetricId": "TotalPciePower",
            "Timestamp": "2020-02-03T20:10:24-06:00",
            "MetricValue": "0.0",
            "Oem": {
                "Dell": {
                    "ContextID": "PowerMetrics",
                    "Label": "PowerMetrics TotalPciePower"
                }
            }
        },
        {
            "MetricId": "TotalStoragePower",
            "Timestamp": "2020-02-03T20:10:24-06:00",
            "MetricValue": "13.2001953125",
            "Oem": {
                "Dell": {
                    "ContextID": "PowerMetrics",
                    "Label": "PowerMetrics TotalStoragePower"
                }
            }
        }
    ],
    "MetricValues@odata.count": 9
}
```