

Dell iDRAC Telemetry

Reference Guide

Notes, cautions, and warnings

 **NOTE:** A NOTE indicates important information that helps you make better use of your product.

 **CAUTION:** A CAUTION indicates either potential damage to hardware or loss of data and tells you how to avoid the problem.

 **WARNING:** A WARNING indicates a potential for property damage, personal injury, or death.

Chapter 1: Overview of iDRAC Telemetry.....	6
Benefits	6
Chapter 2: Terms and definitions.....	7
Chapter 3: Key Features.....	8
Chapter 4: Basic Telemetry Workflow.....	9
Chapter 5: Configuring Telemetry.....	10
Enabling Telemetry Service.....	10
Enabling Global Telemetry though Redfish.....	10
Enabling Global Telemetry Through RACADM.....	11
Enabling Telemetry Streaming Through iDRAC UI.....	11
Disabling Telemetry Service.....	11
Disabling Global Telemetry though Redfish.....	11
Disabling Global Telemetry through RACADM.....	12
Disabling Global Telemetry through iDRAC UI.....	12
Chapter 6: Configuring Metric Report Definitions.....	13
Enabling Metric Report Definition	14
Enabling Metric Report Definition Using Redfish.....	14
Enabling Metric Report Definition through iDRAC UI.....	14
Disabling Metric Report Definition.....	15
Disabling Metric Report Definition using Redfish.....	15
Disabling Metric Report Definition Using iDRAC UI.....	15
Updating Metric Report Definition.....	15
Deleting Metric Report Definition.....	16
Importing and Exporting the Metric Report Definition (using iDRAC UI).....	16
Import and edit the Metric Report Definition.....	16
Export Metric Report Definition	16
Chapter 7: Custom Telemetry Reports.....	17
Create a New Custom Report	17
Replace Custom Report Definition.....	18
Update Custom Report Definition.....	20
Delete Custom Report Definition.....	20
Chapter 8: Receiving Telemetry Reports.....	21
Pulling Reports on Demand.....	21
Pulling a Single Report.....	21
Pulling a Report Collection (URI list only).....	21
Streaming reports.....	22
Server-Sent Events (SSE).....	22

Managing Subscriptions.....	23
Creating subscriptions.....	23
Viewing a single subscription.....	24
Viewing Current Subscriptions.....	24
Deleting a subscription.....	25
Chapter 9: Metric Report Definitions (MRD).....	26
Configurable Properties in MRD.....	26
Read-Only Properties	29
Receiving Telemetry Reports.....	30
Sample Metric Report Definitions.....	30
Report with Temperature Reading of CPU Device.....	30
Report with Temperature and RPM Readings of FAN device.....	31
Report with System Usage Metrics.....	33
Chapter 10: Metric Report.....	37
Properties in a Metric Report.....	37
List of Predefined Metric Reports Supported in Telemetry.....	37
Chapter 11: Metric Definition.....	41
Sample Metric Definition SystemMaxPowerConsumption.....	41
Metrics.....	42
Network Devices.....	42
CPU and Memory Metrics.....	48
Chassis Level and Environmental Metrics.....	55
Accelerators.....	59
PCIe Metrics.....	65
Storage Devices.....	66
System and Platform-Level Monitoring.....	72
Serial Log.....	72
Chapter 12: Metric Properties.....	73
Chapter 13: Triggers.....	74
Configuring Telemetry Report Triggers Through Redfish.....	74
To View All the List of Triggers.....	74
To View Trigger Link To Report.....	75
Setting a Report Trigger.....	75
Configuring Triggers Through GUI.....	76
Import Triggers	76
Export Triggers.....	76
Custom Triggers	77
Creating a New Trigger.....	77
Replacing a Trigger.....	78
Updating a Trigger.....	79
Deleting a Trigger	80
Properties of Triggers.....	80
List of Pre-Defined Triggers.....	81
Sample Triggers Examples.....	81

Chapter 14: Metric Injection.....	83
Enabling Metric Injection Feature.....	83
Enabling metric injection feature using RACADM.....	83
Enable metric injection using Redfish	83
Disabling Metric Injection Feature.....	84
Disable metric injection using RACADM.....	84
Disable metric injection using Redfish.....	84
View current status of metric injection.....	84
OS Metrics	84
Chapter 15: Server Configuration Profile (SCP)	88
Telemetry Configuration Using Server Configuration Profile (SCP).....	88
SCP Export.....	88
SCP Export using Redfish	90
SCP Export using Remote RACADM	91
SCP Export using UI.....	91
SCP Import.....	91
SCP Import using Redfish.....	91
SCP Import using Remote RACADM.....	92
SCP Import using iDRAC UI.....	92
Chapter 16: Historical Temperature Sensor Reports.....	93
Getting Historical Temperature Sensor Reports Through SupportAssist Collector.....	93
Getting Historical Temperature Sensor Reports Through Redfish	93
Chapter 17: Telemetry Reference Tools.....	95
Chapter 18: Report Generation Behavior and Limitations	96
Feature Constraints.....	96
iDRAC10 Telemetry Feature Changes and Limitations.....	97
Behavior of Telemetry feature.....	97
Chapter 19: Best practices.....	98
Chapter 20: Troubleshooting Tips	99
Telemetry Error Messages.....	100
SCP Related Possible Issues.....	102
Missing License error	102
Telemetry Not Enabled	102
Schema Validation Error	103
Chapter 21: Technical support and resources.....	104
Chapter 22: Getting help.....	105
Contacting Dell.....	105

Overview of iDRAC Telemetry

Telemetry is a feature for collecting and streaming live system data from PowerEdge servers to a centralized monitoring service. It supports on-demand data collection and includes metrics such as server performance, health, power, and thermal data. Telemetry requires an iDRAC Datacenter license and is supported on 14G or newer generation platforms.

Telemetry is a one-to-many solution for collecting and streaming live system data from one or more PowerEdge servers to a centralized Remote Server Monitoring, Analysis, and Alerting service. This feature also supports on-demand data collection.

Telemetry data includes metrics such as server and storage performance, health, power, and thermal status. The data can be either streamed (pushed) or collected (pulled) from iDRAC by remote consumers such as Redfish clients.

Telemetry is provided as a granular, time-series report that is streamed or pushed. These telemetry reports are also included in the SupportAssist collection. Data collection and reporting are based on predefined and custom Metric Report Definitions (MRDs) and triggers.

Telemetry streaming settings can be configured using the iDRAC web interface, RACADM, Redfish, or Server Configuration Profile (SCP).

NOTE:

- Telemetry feature is supported on Dell 14th generation (14G) of PowerEdge servers or newer and requires an iDRAC Datacenter license.
- Current Telemetry subscription information is not available through RACADM command on iDRAC10 but is available on iDRAC9.
- There may be differences in how commands are performed in iDRAC9 compared to iDRAC10. Any such variations are highlighted throughout the documentation.

Topics:

- [Benefits](#)

Benefits

Telemetry provides real-time visibility into server health, collecting data on key metrics, enabling IT teams to identify potential issues and optimize server operation.

Dell iDRAC's telemetry feature provides real-time visibility into server health by continuously collecting data on key metrics such as temperature, power usage, fan speed, and CPU performance. This live monitoring enables IT teams to quickly identify potential issues before they become critical, helping to reduce downtime and enhance system reliability. By supporting proactive management, telemetry helps ensure that server environments remain stable, efficient, and well-maintained.

In addition, iDRAC Telemetry supports efficient data center operations by providing insights that help optimize power consumption and cooling. The data is delivered in the industry-standard Redfish format, making it easy to integrate with monitoring tools such as Grafana, Splunk, and Prometheus. This enables users to create custom dashboards, configure alerts, and perform analytics that drive smarter decision-making and cost savings.

Terms and definitions

Introduces essential concepts that are related to monitoring and managing systems.

- **Telemetry Report:** A Telemetry report is a DMTF Redfish Telemetry Specification compliant JSON document. It describes time-series data consisting of metric names, metric values, and timestamps.
- **Metric Report Definition (MRD):** Metric Report Definitions (MRD) is a DMTF-compliant JSON document that describes how a Telemetry Report should be generated and what content it should include. The report output is controlled by the MRD, which contains a list of Metric IDs to include, along with overall configuration details about the report's behavior.
- **Metric Definition (MD):** Metric Definition (MD) is a DMTF-compliant JSON document that describes the attributes of a metric, such as its ID, description, datatype, units, sensing interval, and more.
- **Trigger:** Trigger is a DMTF compliant JSON document defining a set of conditions and associated Metric Reports that should be generated and streamed. The conditions can include a system event or a user-defined condition, such as a metric value crossing a threshold limit or equal to a discrete value.
- **SSE:** Server-Sent Events(SSE) allows a client to open a web service connection to iDRAC, enabling telemetry data to be continuously pushed to the client as needed.
- **SupportAssist Collection:** Dell SupportAssist Collection is a feature that gathers system diagnostics, hardware logs, and configuration data from Dell PowerEdge servers. It helps with proactive troubleshooting, issue detection, and streamlined support case creation.
- **iSM (Integrated Service Module)** is a set of tools for monitoring server health, performance, and configurations. It collects metrics, manages firmware, and integrates with SupportAssist for proactive issue resolution.

Key Features

Describes core capabilities such as customizable reports and triggers, flexible telemetry configuration and streaming, historical data access, and integration with SupportAssist and OS-level metrics.

1. Predefined and customizable MRDs and triggers
2. Telemetry report configuration using Redfish, Web UI, and SCP
3. Stream Telemetry reports using SSE and Post to Subscription methods
4. Pull Telemetry reports on demand.
5. Historical sensor data in Dell SupportAssist collection and pull using Redfish interface
6. Include current Telemetry reports in the Dell SupportAssist collection
7. Process Metric Injection from OS/iSM to include operating system level metrics in Telemetry reports

Basic Telemetry Workflow

The following outlines the steps to configure and enable telemetry reporting on iDRAC, including license installation, global settings, report parameters, and data subscription.

Steps

1. **Install Datacenter license**, if not installed already. ([Datacenter License](#), [Import the license](#))
2. **Configure global telemetry settings**, including enabling the Telemetry service, using RACADM, Redfish, SCP, or the iDRAC web interface.
3. Configure the following telemetry report streaming parameters on the required report using either RACADM, Redfish, or the iDRAC web interface:
 - **EnableTelemetry**
 - **ReportInterval** – to specify a custom interval in place of the default.
 - **ReportTriggers** – optional.
4. Create a subscription request to the Redfish EventService on iDRAC or make an SSE connection to iDRAC.
5. iDRAC generates and pushes the metric report data to the subscribed client periodically or when the defined trigger conditions are met.
6. Alternatively, the Redfish client can pull reports on demand.

Configuring Telemetry

By default, the Telemetry service and individual predefined reports are disabled. To configure Telemetry, the user must enable the Telemetry service and activate the required reports. Additionally, if streaming telemetry data is required, the user must configure the appropriate streaming destinations (subscriptions) to ensure that the data is delivered as expected.

Typically, reports are streamed based on a configured `ReportInterval` condition, but they can also be streamed under error or warning conditions that are known as Triggers, if configured. Trigger definitions are based on iDRAC lifecycle events that are generated for error and warning conditions, or based on user-defined numeric and discrete metric conditions.

Setting	Description
EnableTelemetry	Enable or disable Telemetry globally.
MetricInjection	Enables or disable Metric Injection.

Topics:

- [Enabling Telemetry Service](#)
- [Disabling Telemetry Service](#)

Enabling Telemetry Service

Telemetry Service can be enabled using RACADM, Redfish, SCP (see [SCP Export](#) and [SCP Import](#)), or iDRAC UI.

Enabling Global Telemetry through Redfish

The following describes how to enable global telemetry functionality using Redfish APIs. It outlines the use of standard URIs and PATCH method to activate telemetry services and configure related attributes for data streaming and monitoring.

Using TelemetryService URI:

```
Command: PATCH
URI: /redfish/v1/TelemetryService
Header: content-type application/json
Body: {"ServiceEnabled": <value>}
```

Example:

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

Using Attribute URI:

```
Command: PATCH
IDRAC9 URI:
/redfish/v1/Managers/iDRAC.Embedded.1/Attributes
IDRAC10 URI:
/redfish/v1/Managers/iDRAC.Embedded.1/Oem/Dell/DellAttributes
/iDRAC.Embedded.1
Header: content-type application/json
Auth: Basic or X auth
Body: {"Attributes": {"Telemetry.1.EnableTelemetry": "Enabled"}}
```

Example: iDRAC9

```
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": {"Telemetry.1.EnableTelemetry": "Enabled"}}'
```

Example: iDRAC10

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

Enabling Global Telemetry Through RACADM

The following explains how to enable and verify global telemetry settings using RACADM commands. It includes instructions for checking the current telemetry status and enabling or disabling telemetry functionality through command-line operations.

Get Telemetry Status

Command:

- `racadm get idrac.telemetry.EnableTelemetry`

Set Telemetry Status

`racadm set iDRAC.Telemetry.enableTelemetry Enabled` – Sets the telemetry status to enabled or disabled.

Enabling Telemetry Streaming Through iDRAC UI

The following explains how to enable telemetry streaming using the iDRAC web interface or iDRAC UI. Once telemetry is activated through the Telemetry Data Stream settings, data streaming begins automatically.

Go to **Configuration > System Settings > Telemetry Streaming > Enabled** It enables the global Telemetry Service.

Disabling Telemetry Service

Telemetry can be disabled using Redfish APIs, RACADM commands, or through the iDRAC web interface (iDRAC UI).

Disabling Global Telemetry through Redfish

Disable global telemetry on your PowerEdge server using Redfish API. Learn how to send a PATCH request to the TelemetryService or Attributes endpoint to disable telemetry data collection.

Through DMTF:

```
Command: PATCH
URI: /redfish/v1/TelemetryService
Header: content-type application/json
Body: {"ServiceEnabled": <value>}
```

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

Through Attributes:

```
Command: PATCH
IDRAC9 URI:
/redfish/v1/Managers/iDRAC.Embedded.1/Attributes
IDRAC10 URI:
```

```
/redfish/v1/Managers/iDRAC.Embedded.1/Oem/Dell/DellAttributes
/iDRAC.Embedded.1
Header: content-type application/json
Auth: Basic or X auth
Body: {"Attributes":{"Telemetry.1.DisableTelemetry": "Disabled"}}
```

For example: iDRAC9

```
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":
{"Telemetry.1.EnableTelemetry": "Disabled"}}'
```

For example: iDRAC10

```
curl -s -k -u <user>:<password> -X PATCH https://<IDRAC_IP>/redfish/v1/Managers/
iDRAC.Embedded.1/Oem/Dell/DellAttributes/iDRAC.Embedded.1 -H 'Content-Type: application/
json' -d '{"Attributes": {"Telemetry.1.EnableTelemetry": "Disabled"}}'
```

Disabling Global Telemetry through RACADM

Disable global telemetry on your PowerEdge server using the RACADM command-line tool. Run the following command to stop telemetry data collection.

```
racadm set iDRAC.Telemetry.enableTelemetry Disabled
```

Disabling Global Telemetry through iDRAC UI

Disable global telemetry on your PowerEdge server through the intuitive iDRAC web interface. This method provides a straightforward and intuitive way to manage your server's telemetry settings.

Go to **Configuration > System Settings > Telemetry Screening** and select **Disabled** in the Telemetry Data Stream list.

Configuring Metric Report Definitions

A Metric Report Definition is a collection of metrics that are included in a telemetry report. These metrics cover various system parameters such as CPU usage, memory usage, power consumption, temperature readings, fan speed, and more. PowerEdge systems are shipped with predefined report definitions that have default configurations for periodic reporting. At a minimum, the required reports should be enabled to stream data at a preconfigured recurrence interval. The list of predefined metric report definitions can be viewed in the iDRAC UI or retrieved using the `MetricReportDefinitions` collection query using Redfish.

Each Metric Report Definition has the following report basic settings:

Setting	Description
EnableTelemetry	Enable or disable Telemetry for a report.
ReportInterval	Specify when reports are pushed.

See the [Configurable Properties in MRD](#) for all available configuration settings using the DMTF method. On iDRAC9, a few settings, such as `EnableTelemetry`, `ReportInterval`, and `ReportTriggers`, can also be configured using RACADM commands or Redfish commands. However, the DMTF method is the preferred approach.

To view the list of available predefined report definitions, perform the following Redfish query:

```
Command: GET
URI: /redfish/v1/TelemetryService/MetricReportDefinitions
Header: content-type application/json
```

For example:

```
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:

```
Command: GET
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<report>
Header: content-type application/json
e.g. <report> = CPUSensor
```

For example:

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

Topics:

- [Enabling Metric Report Definition](#)
- [Disabling Metric Report Definition](#)
- [Updating Metric Report Definition](#)
- [Deleting Metric Report Definition](#)
- [Importing and Exporting the Metric Report Definition \(using iDRAC UI\)](#)

Enabling Metric Report Definition

A Metric Report Definition can be enabled through the iDRAC web interface (iDRAC UI) or by using the Redfish API.

Enabling Metric Report Definition Using Redfish

The following explains how to enable a Metric Report Definition and configure its recurrence interval using Redfish APIs. It includes example PATCH commands for activating telemetry reports and setting custom reporting intervals.

The below URL is applicable for both iDRAC9 and iDRAC10.

```
Command: PATCH
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<report>
Header: content-type application/json
Body: {"MetricReportDefinitionEnabled": true}
e.g. <report> = CPUSensor
```

For example:

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

To configure a report recurrence interval if different from default (for example, 2 minutes):

```
Command: PATCH
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<report>
Header: content-type application/json
Body: {"Schedule": {"RecurrenceInterval": "PT0H2M0S"}}
```

For example:

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

Enabling Metric Report Definition through iDRAC UI

The following outlines the steps to enable a Metric Report Definition using the iDRAC web interface (iDRAC UI). You can access telemetry settings, edit report properties, and configure the report state and other parameters through the iDRAC UI.

About this task

Steps

1. Go to **Configuration > System Settings > Telemetry Configuration > Metric Report Definition..**
2. Go to **Actions** dropdown and select **Edit Report properties.**
3. Configure the metric report by selecting **Enabled** in Metric Report State and then update the remaining settings as needed.

 **NOTE:** For more information about how to use the iDRAC UI, click Help on the right corner of the iDRAC UI page. For more information about Metric Report Properties, see [Configurable Properties in MRD](#).

Disabling Metric Report Definition

Disabling a Metric Report Definition can be done through the Redfish API or the iDRAC user interface. This process stops the generation of specific system metrics reports, helping to manage system resources or align with updated monitoring policies.

Disabling Metric Report Definition using Redfish

Disabling a metric report definition using Redfish API involves sending a PATCH request with a JSON payload to update the report's enabled status.

```
Command: PATCH
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<report>
Header: content-type application/json
Body: {"MetricReportDefinitionEnabled": false}
e.g. <report> = PowerMetrics
```

For example:

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

Disabling Metric Report Definition Using iDRAC UI

The following outlines the steps to disable a Metric Report Definition using the iDRAC web interface (iDRAC UI).

Steps

1. Go to **Configuration > System Settings > Telemetry Configuration > Metric Report Definition**.
2. Go to **Actions** dropdown > Select **Edit Report properties**.
3. Disable the metric report by selecting **"Disabled"** in Metric Report State and then update the remaining settings as needed.

 **NOTE:** For more information about how to use the iDRAC UI, click Help on the right corner of the iDRAC UI page. For more information about Metric Report Properties, see [Configurable Properties in MRD](#).

Updating Metric Report Definition

Metric Report Definitions (MRDs) can be updated using the Redfish API by sending a PATCH request to the specific report URI. The Predefined Metric Report Definitions can be updated as needed.

```
Command: PATCH
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<report>
Header: content-type application/json
Body: {"Schedule": {"RecurrenceInterval": "<value>"}, MetricReportDefinitionType":
"<value>"}
```

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. Example :
<MRD> = FanSensor

 **NOTE:** To restore predefined reports to their default values, delete the customized report definition (MRD), such as PowerMetrics.

Deleting Metric Report Definition

To delete a Metric Report Definition (MRD) using the Redfish API, send a DELETE request to the specific MRD URI.

```
Command: DELETE
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<MRD>
Header: content-type application/json
e.g. <report> = FanSensor
```

For example:

```
curl -s -k -u <user>:<password> -X DELETE https://<IDRAC_IP>/redfish/v1/TelemetryService/MetricReportDefinitions/FanSensor
```

 **NOTE:** If predefined reports are deleted, the system automatically recovers them within a few seconds. For more information, see [List of Predefined Metric Reports Supported in Telemetry](#).

Importing and Exporting the Metric Report Definition (using iDRAC UI)

You can import or export Metric Report Definitions (MRDs) using the iDRAC web interface. See to the following sections for detailed instructions.

Import and edit the Metric Report Definition

To customize a Metric Report Definition (MRD), follow these steps to import and edit it using the iDRAC web interface:

Steps

1. Go to **Configuration > System Settings > Telemetry Configuration > Metric Report Definition**.
2. Select the **Location** Type.
3. Click **Choose File** and **Select the file**.
4. Click **Import**.. The metrics report file is imported. The report is displayed in the Telemetry reports list.
5. To edit a specific report, click **Actions > Edit Report Properties**. The Report Settings dialog box is displayed.
6. Edit the report settings as needed and click **Save**.

Export Metric Report Definition

Export a Metric Report Definition (MRD) to compare server performance or use it as a template for other servers.

Steps

1. Go to **Configuration > System Settings > Telemetry Configuration**.
2. Select the **Location** Type.
3. Select the **Metric Report Definition**..
4. Click **Save**. The metrics report file is saved.

 **NOTE:** For sample MRDs to import, see [Sample Metric Report Definitions](#). Alternatively, export an existing MRD to view its structure and use it as a reference.

Custom Telemetry Reports

Not all users are interested in all the predefined metric and properties. iDRAC Telemetry solution includes a large set of metrics which are grouped in pre-defined MRDs. There comes need to create 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, and so forth, independently. And the metric reports can be customized by selecting the needed arbitrary metrics in the metric report definition. The existing predefined reports can also be modified or configured with wanted metrics and properties, but that would impact all clients that use the predefined report. For the detailed explanation of all available MRD properties, see [Configurable Properties in MRD](#).

Topics:

- [Create a New Custom Report](#)
- [Replace Custom Report Definition](#)
- [Update Custom Report Definition](#)
- [Delete Custom Report Definition](#)

Create a New Custom Report

Below is an example to post a custom MRD with specific properties using Redfish interface. Typically, one can GET any existing Predefined 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 wanted NIC port (FQDD) - NIC.Slot.1-1-1 is requested.

```
Command: POST
URI: /redfish/v1/TelemetryService/MetricReportDefinitions
Header: content-type application/json
BODY: <custom MRD definition json>
```

For example:

```
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 is successful the new custom report is added to the report definition collection. The report collection shall contain both custom metric reports along with Predefined metric reports.

To get report definition collection (URI list only) which include custom report definition:

```

Command: GET
URI: /redfish/v1/TelemetryService/MetricReportDefinitions
Header: content-type application/json

```

For example:

```

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

```

To get one report definition detail:

```

Command: GET
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<report>
Header: content-type application/json
e.g. <report> = TxRxBytesNicSlot1

```

For example:

```

curl -s -k -u <user>:<password> -X GET https://<IDRAC_IP>/redfish/v1/TelemetryService/
MetricReports/TxRxBytesNicSlot1.

```

Replace Custom Report Definition

If the Metric Report Definition exists, it is replaced with the new definition.

```

Command: PUT
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<custom_report>
Header: content-type application/json
For e.g. <custom_report> = TxRxBytesNicSlot1
Body: <custom MRD definition json>

```

For example:

```
curl -s -k -u <user>:<password> -X PUT https://<IDRAC_IP>/redfish/v1/TelemetryService/
MetricReportDefinitions/TxRxBytesNicSlot1 -H 'Content-Type: application/json' -d '{
  "Id": "TxRxBytesNicSlot1",
  "Name": "Tx, Rx, Errors, and Dropped Packets from NIC Slot 1",
  "Description": "Metric report updated to include Tx, Rx, Packet Errors, and Dropped
Packets for NIC Slot 1 at 5-minute intervals",
  "MetricReportDefinitionEnabled": true,
  "MetricReportDefinitionType": "Periodic",
  "MetricReportHeartbeatInterval": "PT0H0M0S",
  "SuppressRepeatedMetricValue": false,
  "ReportTimespan": "PT0H0M0S",
  "ReportUpdates": "Overwrite",
  "ReportActions": ["redfishEvent"],
  "Schedule": {
    "RecurrenceInterval": "PT0H5M0S"
  },
  "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
        }
      }
    },
    {
      "MetricId": "PacketErrors",
      "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": "DroppedPackets",
      "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": 4,
  "Links": {
    "Triggers": []
  }
}

```

Update Custom Report Definition

The custom report definitions can be updated as needed.

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

```

Command: PATCH
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<MRD>
Header: content-type application/json
BODY: {"Schedule": {"RecurrenceInterval": "PT0H5M0S"}, MetricReportDefinitionType":
"OnRequest"}
e.g. <MRD> = TxRxBytesNicSlot1

```

For example:

```

curl -s -k -u <user>:<password> -X PATCH https://<IDRAC_IP>/redfish/v1/
TelemetryService/MetricReportDefinitions/TxRxBytesNicSlot1 -H 'Content-Type: application/
json' -d '{"Schedule": {"RecurrenceInterval": "PT0H5M0S"}, MetricReportDefinitionType":
"OnRequest"}'

```

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

Delete Custom Report Definition

To delete a Report Definition:

```

Command: DELETE
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<report>
Header: content-type application/json
e.g. <report> = TxRxBytesNicSlot1

```

For example:

```

curl -s -k -u <user>:<password> -X DELETE https://<IDRAC_IP>/redfish/v1/TelemetryService/
MetricReportDefinitions/TxRxBytesNicSlot1

```

Receiving Telemetry Reports

After Telemetry streaming is configured on the iDRAC, Redfish clients can either stream Telemetry reports continuously or pull Telemetry reports on demand. The following sections describe the methods through which the clients can receive the data.

Topics:

- [Pulling Reports on Demand](#)
- [Streaming reports](#)
- [Managing Subscriptions](#)

Pulling Reports on Demand

An API client can retrieve Telemetry reports at any time by performing an HTTP GET operation on the specified URIs.

Pulling a Single Report

To pull a single report, use the following HTTP GET operation:

```
Command: GET
URI: /redfish/v1/TelemetryService/MetricReports/<report>
Header: content-type application/json
e.g. <report> = PowerMetrics
```

For example:

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

Pulling a Report Collection (URI list only)

To retrieve a list of available report URIs, use the following HTTP GET operation:

```
Command: GET
URI: /redfish/v1/TelemetryService/MetricReports
Header: content-type application/json
```

This returns a list of available report URIs that can be used to pull individual reports.

For example:

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

Streaming reports

Streaming reports provide real-time data and insights, available in various formats, including telemetry streaming for automated communications and data collection from remote points, with options for on-demand or scheduled reporting.

Server-Sent Events and Subscriptions are two ways in which metric reports can be continuously streamed. When either SSE connection is made, or Subscription is posted an entry is added to the subscription list.

There can be a total of 8 entries in the list (including both types). When the connection closes, or subscription is removed from the list. When one of the SSE connections is closed or Subscription is removed, then the subscription is removed from the active subscription list.

Server-Sent Events (SSE)

Server-Sent Events allow a client to receive automatic updates from a server, enabling real-time data streaming and efficient communication.

The Server-Sent Events (SSE) method is another way to stream Telemetry data from the iDRAC to an API client. This method uses the HTML5 SSE feature and the HTTP protocol to communicate between the client and the iDRAC event service.

How SSE works

Server-Sent Events enable real-time data streaming, allowing clients to receive automatic updates from servers for efficient communication and telemetry data transfer.

When a client performs a GET request on the SSE URI, the iDRAC event service establishes a connection and starts streaming enabled metric reports to the client. The client can also provision the SSE URI to query specific metric reports using the \$filter parameter. The response depends on the EventFormatType in the \$filter parameter. All responses are Redfish compatible.

Streaming all reports

The SSE URI contains the event format type, which is set to "Metric Report". This directs the iDRAC event service to stream only enabled metric reports.

URI for iDRAC9:

```
Command: GET
iDRAC9 URI:
/redfish/v1/SSE?$filter=EventFormatType%20eq%20MetricReport
iDRAC10
URI:/redfish/v1/SSE?%24filter=EventFormatType%20eq%20%27MetricReport%27
Header: content-type application/json
```

For example: iDRAC9

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

For example: iDRAC10

```
curl -N -k -u <user>:<password> -X GET
'https://<IDRAC_IP>/redfish/v1/SSE?
%24filter=EventFormatType%20eq%20%27MetricReport%27'
```

Streaming a single report

To stream a single report, the client can use the \$filter parameter to specify the metric report definition.

```
Command: GET
URI: /redfish/v1/SSE?$filter=MetricReportDefinition%20eq%20%27/redfish/v1/
TelemetryService/MetricReportDefinitions/<report>%27
```

```
Header: content-type application/json
e.g. <report> = FanSensor
```

For example:

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

Connection Termination

The connection can be terminated by either the client or the iDRAC event service. If there is no Telemetry data sent to the client for more than an hour, the connection is closed from the event service endpoint. If the connection drops off due to a network glitch or unknown reasons, the last event ID is provided by the client to the event service to resume streaming.

Managing Subscriptions

An API client can receive Telemetry reports using the subscription method. This method involves creating a subscription specifying the destination using an 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 through an HTTP POST request. The API client can receive reports by listening on the destination port.

NOTE:

- Max Configuration for Telemetry data - No restriction (All 50 reports including Custom MRDs to 8 subscription destinations).
- Min Configuration for Telemetry data - Only pre-canned reports and max 2 subscriptions.

Creating subscriptions

To create a subscription, the client sends an HTTP POST request to the iDRAC Event Service. The request includes the following parameters:

Table 1. HTTP POST request parameters

Parameters	Description
Destination	The IP address and port number where the client will receive the Telemetry reports.
EventFormatType	The type of event format, which is set to "MetricReport" for Telemetry reports.
Context	A name for the subscription.
Protocol	The protocol used to receive the reports, which is set to "Redfish".
EventTypes	The types of events to receive, which is set to ["MetricReport"] for Telemetry reports.
SubscriptionType	The type of subscription, which is set to "RedfishEvent".
MetricReportDefinitions	An array of metric report definitions to receive, if list not specified all enabled reports will be received.

The following example create a new subscription with subscription id. To get the subscription id, see [Viewing Current Subscriptions](#) where you can view all the subscriptions.

```
Command: POST
URI: https://<IDRAC_IP>/redfish/v1/EventService/Subscriptions
Body: {
```

```
"Destination": "https://<listener ip:port>",
"EventFormatType": "MetricReport",
"Context": "TelmetryTest",
"Protocol": "redfish",
"EventTypes": ["MetricReport"],
"SubscriptionType": "redfishEvent"
}
Header: content-type application/json
```

For example:

```
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/" },
"@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/"}]}'"
```

The details of each parameter can be found here [Parameter details](#).

Viewing a single subscription

To view a subscription, the client sends an HTTP DELETE request to the iDRAC Event Service, specifying the subscription ID.

```
Command: GET
URI: https://<IDRAC_IP>/redfish/v1/EventService/Subscriptions/<subscription-id>
Header: content-type application/json
```

For example:

```
curl -s -k -u <user>:<password> -X GET https://<IDRAC_IP>/redfish/v1/EventService/
Subscriptions/<subscription-id> -H 'Content-Type: application/json'
```

You can replace the subscription-id with actual subscriptionid, which was generated when you created a new subscription before executing the example curl command.

Viewing Current Subscriptions

To view the current subscriptions, the client can send an HTTP GET request to the iDRAC Event Service.

```
Command: GET
URI: https://<IDRAC_IP>/redfish/v1/EventService/Subscriptions
Header: content-type application/json
```

For example:

```
curl -s -k -u <user>:<password> -X GET https://<IDRAC_IP>/redfish/v1/EventService/
Subscriptions -H 'Content-Type: application/json'
```

Deleting a subscription

To delete a subscription, the client sends an HTTP DELETE request to the iDRAC Event Service, specifying the subscription ID. Clients should terminate subscriptions by sending this request.

```
Command: DELETE
URI: https://<IDRAC_IP>/redfish/v1/EventService/Subscriptions/<subscription-id>
Header: content-type application/json
```

Replace <IDRAC_IP> with the IP address of the iDRAC and < subscription-id > with the ID of the subscription that must be deleted.

For example:

```
curl -s -k -u <user>:<password> -X DELETE https://<IDRAC_IP>/redfish/v1/EventService/
Subscriptions/<subscription-id> -H 'Content-Type: application/json'
```

Metric Report Definitions (MRD)

Metric Report Definitions (MRD) are the foundation of Telemetry metric reports. They define the properties and configuration options for generating and streaming Telemetry reports. The DMTF Redfish compliant spec [Redfish Resource and Schema Guide](#) for MRDs provides a standardized way to define and configure Telemetry reports.

Topics:

- [Configurable Properties in MRD](#)
- [Read-Only Properties](#)
- [Sample Metric Report Definitions](#)

Configurable Properties in MRD

- `MetricReportDefinitionEnabled` – This property is used to enable or disable the report. The default is “false.”
- `MetricReportDefinitionType` – Defines the type of metric report that is generated from the metric report definition. The below table defines the different `MetricReportDefinitionType(s)` and their meanings.

Table 2. MetricReportDefinitionType(s) and their meanings.

MetricReportDefinitionType(s)	Meaning
Periodic	This is the default for any new reports that are uploaded with an unspecified type. This type of report must have a <code>Schedule</code> object with a <code>RecurrenceInterval</code> property. The report is repeatedly generated on a periodic basis as requested in the “<code>Schedule/RecurrenceInterval</code>” property, with the report being generated at the expiry of the time period specified by <code>RecurrenceInterval</code> property. When metric reports are generated, the report includes metrics greater than the previous report timestamp up to and including the new report timestamp. If a <code>ReportTimespan</code> is also present, the resulting metric value list includes the set-wise union of values that are covered by <code>ReportTimespan</code> and metric values with timestamps in the interval of previous <code>ReportTimestamp</code> to current <code>ReportTimestamp</code>.
OnChange	This report is generated anytime a referenced metric changes. To prevent overloading several subsystems with reports, the minimum time between reports is 10 seconds. If multiple metrics are changed during the 10 seconds “cooling off” period, all metrics that have changed up until the report generation time are in the report. For <code>OnChange</code> reports, a non-null <code>ReportTimespan</code> property is mandatory, the <code>SuppressRepeatedMetricValue</code> property is required to be set to true. The report shall contain all metrics values newer than <code>ReportTimestamp</code> – <code>ReportTimespan</code> up until the <code>ReportTimestamp</code>. Also, <code>OnChange</code> requires <code>MetricReportHeartBeatInterval</code> and <code>schedule</code> to be cleared.
OnRequest	This report is generated when there is an HTTP GET in the report. The report contains all metrics values newer than <code>Now()</code> - <code>ReportTimeSpan</code>. Other Requirements:

Table 2. MetricReportDefinitionType(s) and their meanings. (continued)

MetricReportDefinitionType(s)	Meaning
	- ReportUpdates field is IGNORED in input. Output is fixed to AppendWrapsWhenFull - ReportActions field is IGNORED in input. Output is fixed to LogToMetricReportsCollection - ReportTimeSpan must have a value greater than 0 - Schedule property must be either not present in input, or the RecurrenceInterval must be 'null' or equivalent to any variant of 'PT0S'.

- Metric Properties - MetricProperties contain references to properties in the redfish tree that the user may incorporate into Telemetry. If the path is not present, the MetricProperties will be kept in case the path will surface in the future with feature enablement or hardware adds. The TelemetryService monitors the values of the properties specified here and adds them to the MetricReport when they are added or changed in Redfish. The MetricReport has an expanded MetricProperty in the array of MetricValues.
- The MRD MetricProperties is an array of strings. Each string contains alphanumeric characters in addition to the characters “-_/#{ }”. Each string specifies the Redfish property URI as per RFC6901. For example, /redfish/v1/Systems/System.Embedded.1/Oem/Dell/DellNumericSensors/iDRAC.Embedded.1_0x23_SystemBoardInletTemp#CurrentReading or NULL. The Redfish Property URI includes the URL. For example, /redfish/v1/Systems/System.Embedded.1/Oem/Dell/DellNumericSensors/iDRAC.Embedded.1_0x23_SystemBoardInletTemp followed by a hash “#” and the JSON property path for example, CurrentReading. Within the imaginative landscape when property B is collected in JSON {“A”:{“B”：“C”}} the property path is “A/B” . The user may replace any part of the URI with a wildcard. A wildcard consists of a text name inside the curly braces, “{systemName}”.
- WildCards – The wildcard(s) work in conjunction with the MetricProperties above. For every MetricProperty that contains a wildcard the TelemetryService substitutes in wildcard value(s) from this property. The wildcards property consists of an array of key/value pairs. Each key matches the name of the wildcard specified in the metricproperties above. Each value is an array of substitutions strings. Wildcards require the variable presence in MetricProperties, otherwise an error is returned. For example,
 - WildCards – [{“Name”: “systemName”, “Values”: [“System.Embedded.1”] },
 - {“Name”: “sensor”, “Values”: [“SystemBoardInletTemp”,
 - “SystemBoardExhaustTemp”] }
- ReportActions – This property specifies what should happen when a report is generated per the MetricReportDefinitionType, above. The property is an ARRAY, containing zero or more of the following settings. If no settings are specified, it disables the report. Default is [“LogToMetricReportsCollection,” “RedfishEvent”].

Table 3. ReportActions and their meanings.

ReportActions	Meaning
LogToMetricReportsCollection	The generated metric report is added to the Metric Reports collection with an “Id” matching the report definition
RedfishEvent	The generated metric report is sent as a “Metric Report” event type to all subscribers: SSE Subscribers and POST Subscribers. The report is not saved to the metric reports collection unless LogToMetricReportsCollection is also specified. NOTE: The individual subscription types may further have facilities to filter which reports are sent. For example, SSE can query only specific reports. This property provides base enablement but does not force this report into streams that have filters.

- ReportUpdates – This property controls how subsequent reports are handled after the first report is generated. The below table defines the different ReportUpdates and their meanings.

Table 4. ReportUpdates and their meanings.

ReportUpdates	Meaning
AppendStopsWhenFull	Each time the report is generated, the new metric values are added to the end of the MetricValues array. The report can contain up to an "AppendLimit" (not user configurable, described below) number of metric values. Once the maximum number of metric values have been added, no more will be added to the report, and the MetricReportDefinition is automatically disabled. The user must PATCH the MetricReportDefinition back to enable the report to restart the report generation. Reports will not stream after they have stopped. The detection of hitting append limit is 'asynchronous', done periodically as a background task, so streaming may not immediately stop, but should stop within 15 minutes of hitting the limit.
AppendWrapsWhenFull	When the report is created, new Metrics are added to the MetricValues array, keeping existing entries. Once there are more than AppendLimit numbers of metric values in the MetricValues array, older entries above the AppendLimit number are dropped from the report.
NewReport	A new report is added, with a name generated taking the report definition name as a base and adding a dash, then the current date or time. Since this implies that many older reports are available under the older names, this can present a challenge to iDRAC, so defining this specific behavior for allows: • Deleting any report that is already processed and is not needed anymore. • Automatic deletion of reports that are older than the last three, as iDRAC will keep at most three completed metric reports per report definition.
Overwrite	Overwrites the older report with a report of the same name containing newer data. This is the default

- **SuppressRepeatedMetricValue**– If enabled, new metrics are added to the Metrics array if they are different from the last reported metric. For a single report, if a metric value does not change, it is not added again. SuppressRepeatedMetricValue prevents the addition of consecutive duplicate metric values to a specific metric value stream. Default is "Disabled" for new custom reports.
- **MetricReportHeartbeatInterval** – For metric reports that have SuppressRepeatedMetricValue set to true, some metrics may not change for a long time and fail to show up in many reports. This is expected, but sometimes end users may want to have periodic 'reminders' (conceptually) of which metrics are valid for this report. The heartbeat setting will periodically force a report that contains at least one value for every valid metric. This property is a Redfish Duration and should always be greater than or equal to the RecurrenceInterval.

This will only unsuppress nonaggregated metrics. Any metrics with a CollectionFunction are not affected by this setting. The way this duration works is that every time that is evenly divisible by the duration constitutes a conceptual trigger point where the 'next' report that is generated after that time will be a heartbeat report. MetricReportHeartbeatInterval is only valid for Periodic type reports. The default is 0 (disabled). Setting to NULL or 0 will disable this feature.

- **Schedule** – This property is the DMTF standard Schedule property and specifies the recurrence for the report. The RecurrenceInterval property of the Schedule object specifies a Redfish Duration string. This property is valid for Periodic reports only. When set, reports are generated at the specified interval.

A RecurrenceInterval value of 0 results in a nonrepeating, single report that may or may not have data depending on the availability of data in the database.

- **ReportTimespan** – This property specifies the duration of the report. This attribute must be set if the report is only required for a specific period or time span. This property setting is not applicable to "SerialLog" report where the report (log) continuity is expected.

- Metrics – An array property that lists the MetricId(s) to be in the metric report which the metric report definition defines. Each Metric can have the following configurable attributes.

Table 5. Metric and Configurable attributes

Metric	Configurable attribute
CollectionFunction	The Redfish standard defines a way to reduce the raw number of data points that are sent by aggregating metrics using predefined metric functions that are applied during collection. The possible values are Average, Maximum, Minimum, and Summation. The default is null. When applying a collection function, the function is applied across a time interval and one single value is added to the report. If specified, CollectionDuration must also be present.
CollectionDuration	Specifies the duration over which the function is computed. Specified according to the Duration format in the Redfish schema supplement. The default is 'null'. If specified, CollectionFunction must also be specified.
CollectionTimeScope	The possible values are - Interval – The timestamp of the resulting MetricValue is the end of the time interval. This is used and only valid when CollectionFunction and CollectionDuration are specified. Point– Metric values are point in time values (implies no collection function applied). This is the default associates.
MetricId	The label for the metric definition that is derived by applying the collectionFunction to the metric property. It matches the Id property of the corresponding metric definition.
MetricProperties	The set of URIs for the properties on which this metric is collected.
OEM/Dell/Source	Allows you to specify the data source to include. This is logically “AND”-ed with the MetricId and FQDD, if specified. It can be specified by itself to request that all MetricIds from that source be included.
OEM/Dell/FQDD	Allows you to specify an FQDD filter. This is logically “AND”-ed with all other selection criteria. It can be specified by itself to request all MetricIds that are reported for that FQDD.
OEM/Dell/CustomLabel	The value of the customLabel field is used as the label for that metric in the metric report and overrides the default label.

Read-Only Properties

The read-only Properties are:

- AppendLimit - This value is a read-only value that is determined by the Telemetry Service and will be added as output to any Metric Report Definition. This parameter represents the maximum number of MetricValues in a report. The default value is 2400.
- Status - This read-only property reflects the current state of the report definition. Values are Enabled or Disabled and reflect the MetricReportDefinitionEnabled value. If the report is AppendStopsWhenFull, then this is updated to Disabled and the MetricReportDefinitionEnabled property.

Receiving Telemetry Reports

The read-only OEM attributes are:

- OEM/Dell/Digest - The attribute allows you to identify out of band changes in custom MRD outside the influence of the component that created it. Digest represents a consistent hash of the editable fields in the metric report definition and remain constant if the definition remains intact. Digest changes with updates in the report definition. Digest also changes with firmware updates if the updated version includes changes to editable metric report definition fields.
- OEM/Dell/iDRACFirmwareVersion - Represents the current installed version of the iDRAC firmware.

Sample Metric Report Definitions

After Telemetry streaming is configured on the iDRAC, Redfish clients can either stream Telemetry reports continuously or pull Telemetry reports on demand. The following sections describe the methods through which clients can receive the data.

Report with Temperature Reading of CPU Device

```
{
  "@odata.type": "#MetricReportDefinition.v1_4_2.MetricReportDefinition",
  "@odata.context": "/
Redfish/v1/$metadata#MetricReportDefinition.MetricReportDefinition",
  "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/CPU Sensor",
  "Id": "CPUSensor",
  "Name": "CPU Sensor Metric Report",
  "Description": "CPU Sensor",
  "AppendLimit": 2400,
  "MetricReportDefinitionEnabled": true,
  "MetricReportDefinitionType": "Periodic",
  "MetricReportHeartbeatInterval": "PT0H0M0S",
  "SuppressRepeatedMetricValue": false,
  "ReportTimespan": "PT0H0M0S",
  "ReportUpdates": "Overwrite",
  "Wildcards": [],
  "MetricReportDefinitionType@redfish.AllowableValues": [
    "Periodic",
    "OnChange",
    "OnRequest"
  ],
  "ReportUpdates@redfish.AllowableValues": [
    "AppendStopsWhenFull",
    "AppendWrapsWhenFull",
    "NewReport",
    "Overwrite"
  ],
  "ReportActions": [
    "RedfishEvent",
    "LogToMetricReportsCollection"
  ],
  "ReportActions@Redfish.AllowableValues": [
    "LogToMetricReportsCollection",
    "RedfishEvent"
  ],
  "Status": {
    "State": "Enabled"
  },
  "Schedule": {
    "RecurrenceInterval": "PT0H1M0S"
  },
  "MetricReport": {
    "@odata.id": "/redfish/v1/TelemetryService/MetricReports/CPUSensor"
  },
  "Metrics": [
    {
      "MetricId": "TemperatureReading",
      "MetricProperties": [],
      "CollectionFunction": null,
    }
  ]
}
```

```

        "CollectionDuration": null,
        "CollectionTimeScope": "Point",
        "Oem": {
            "Dell": {
                "@odata.type": "#DellMetric.v1_1_0.DellMetric",
                "CustomLabel": null,
                "FQDD": "CPU.Socket.%",
                "Source": null
            }
        }
    },
    "Links": {
        "Triggers": [
            {
                "@odata.id": "/redfish/v1/TelemetryService/Triggers/CPUCriticalTrigger"
            },
            {
                "@odata.id": "/redfish/v1/TelemetryService/Triggers/CPUWarnTrigger"
            },
            {
                "@odata.id": "/redfish/v1/TelemetryService/Triggers/
TMPCpuCriticalTrigger"
            },
            {
                "@odata.id": "/redfish/v1/TelemetryService/Triggers/TMPCpuWarnTrigger"
            }
        ]
    },
    "Oem": {
        "Dell": {
            "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
            "Digest": "0f66e7f33aabe9707728c0b9297d170697df27105f7a672373c131ea8baf9481",
            "iDRACFirmwareVersion": "1.20.25.00"
        }
    }
}

```

Report with Temperature and RPM Readings of FAN device

```

{
    "@odata.type": "#MetricReportDefinition.v1_3_3.MetricReportDefinition",
    "@odata.context": "/
redfish/v1/$metadata#MetricReportDefinition.MetricReportDefinition",
    "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/
TemperatureAndRPMReading",
    "Id": "TemperatureAndRPMReading",
    "Name": "All Temperature and RPM Readings",
    "Description": "A report with all relevant thermal metrics - Temperature reading and
FAN RPM readings",
    "AppendLimit": 2400,
    "MetricReportDefinitionEnabled": true,
    "MetricReportDefinitionType": "Periodic",
    "MetricReportHeartbeatInterval": "PT0H0M0S",
    "SuppressRepeatedMetricValue": false,
    "ReportTimespan": "PT0H1M0S",
    "ReportUpdates": "Overwrite",
    "MetricReportDefinitionType@Redfish.AllowableValues": [
        "Periodic",
        "OnChange",
        "OnRequest"
    ],
    "MetricReportDefinitionType@Redfish.AllowableValues@odata.count": 3,
    "ReportUpdates@Redfish.AllowableValues": [
        "AppendStopsWhenFull",
        "AppendWrapsWhenFull",
        "NewReport",
        "Overwrite"
    ]
}

```

```

    ],
    "ReportUpdates@Redfish.AllowableValues@odata.count": 4,
    "ReportActions": [
        "LogToMetricReportsCollection",
        "RedfishEvent"
    ],
    "ReportActions@odata.count": 2,
    "ReportActions@Redfish.AllowableValues": [
        "LogToMetricReportsCollection",
        "RedfishEvent"
    ],
    "ReportActions@Redfish.AllowableValues@odata.count": 2,
    "Status": {
        "State": "Enabled"
    },
    "Wildcards": [],
    "Wildcards@odata.count": 0,
    "Schedule": {
        "RecurrenceInterval": "PT0H1M0S"
    },
    "MetricReport": {
        "@odata.id": "/Redfish/v1/TelemetryService/MetricReports/
TemperatureAndRPMReading"
    },
    "Metrics": [
        {
            "MetricId": "TemperatureReading",
            "MetricProperties": [],
            "MetricProperties@odata.count": 0,
            "CollectionFunction": null,
            "CollectionDuration": null,
            "CollectionTimeScope": "Point",
            "Oem": {
                "Dell": {
                    "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
                    "CustomLabel": null,
                    "FQDD": null,
                    "Source": null
                }
            }
        },
        {
            "MetricId": "RPMReading",
            "MetricProperties": [],
            "MetricProperties@odata.count": 0,
            "CollectionFunction": null,
            "CollectionDuration": null,
            "CollectionTimeScope": "Point",
            "Oem": {
                "Dell": {
                    "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
                    "CustomLabel": null,
                    "FQDD": null,
                    "Source": null
                }
            }
        }
    ],
    "Metrics@odata.count": 2,
    "Links": {
        "Triggers": [],
        "Triggers@odata.count": 0
    },
    "Oem": {
        "Dell": {
            "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
            "Digest": "82f47e45c966bdbdf33cd904756178f934fb86ec55d4dfcdae2c0b8e129be2d8",
            "iDRACFirmwareVersion": "5.10.00.00"
        }
    }

```

```
}
}
```

Report with System Usage Metrics

```
{
  "@odata.type": "#MetricReportDefinition.v1_3_3.MetricReportDefinition",
  "@odata.context": "/Redfish/v1/$metadata#MetricReportDefinition.MetricReportDefinition",
  "@odata.id": "/Redfish/v1/TelemetryService/MetricReportDefinitions/SystemUsageAggregations",
  "Id": "SystemUsageAggregations",
  "Name": "System usage metrics",
  "Description": "A report with average, maximum and minimum system usage metrics",
  "AppendLimit": 2400,
  "MetricReportDefinitionEnabled": true,
  "MetricReportDefinitionType": "Periodic",
  "MetricReportHeartbeatInterval": "PT0H0M0S",
  "SuppressRepeatedMetricValue": false,
  "ReportTimespan": "PT0H1M0S",
  "ReportUpdates": "Overwrite",
  "MetricReportDefinitionType@Redfish.AllowableValues": [
    "Periodic",
    "OnChange",
    "OnRequest"
  ],
  "MetricReportDefinitionType@Redfish.AllowableValues@odata.count": 3,
  "ReportUpdates@Redfish.AllowableValues": [
    "AppendStopsWhenFull",
    "AppendWrapsWhenFull",
    "NewReport",
    "Overwrite"
  ],
  "ReportUpdates@Redfish.AllowableValues@odata.count": 4,
  "ReportActions": [
    "LogToMetricReportsCollection",
    "RedfishEvent"
  ],
  "ReportActions@odata.count": 2,
  "ReportActions@Redfish.AllowableValues": [
    "LogToMetricReportsCollection",
    "RedfishEvent"
  ],
  "ReportActions@Redfish.AllowableValues@odata.count": 2,
  "Status": {
    "State": "Enabled"
  },
  "Wildcards": [],
  "Wildcards@odata.count": 0,
  "Schedule": {
    "RecurrenceInterval": "PT0H1M0S"
  },
  "MetricReport": {
    "@odata.id": "/Redfish/v1/TelemetryService/MetricReports/SystemUsageAggregations"
  },
  "Metrics": [
    {
      "MetricId": "CPUUsage",
      "MetricProperties": [],
      "MetricProperties@odata.count": 0,
      "CollectionFunction": "Average",
      "CollectionDuration": "PT0H1M0S",
      "CollectionTimeScope": "Interval",
      "Oem": {
        "Dell": {
          "@odata.type": "#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
          "CustomLabel": null,
          "FQDD": null,
          "Source": null
        }
      }
    }
  ]
}
```

```

    }
  },
  {
    "MetricId": "CPUUsage",
    "MetricProperties": [],
    "MetricProperties@odata.count": 0,
    "CollectionFunction": "Maximum",
    "CollectionDuration": "PT0H1M0S",
    "CollectionTimeScope": "Interval",
    "Oem": {
      "Dell": {
        "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
        "CustomLabel": null,
        "FQDD": null,
        "Source": null
      }
    }
  },
  {
    "MetricId": "CPUUsage",
    "MetricProperties": [],
    "MetricProperties@odata.count": 0,
    "CollectionFunction": "Minimum",
    "CollectionDuration": "PT0H1M0S",
    "CollectionTimeScope": "Interval",
    "Oem": {
      "Dell": {
        "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
        "CustomLabel": null,
        "FQDD": null,
        "Source": null
      }
    }
  },
  {
    "MetricId": "IOUsage",
    "MetricProperties": [],
    "MetricProperties@odata.count": 0,
    "CollectionFunction": "Average",
    "CollectionDuration": "PT0H1M0S",
    "CollectionTimeScope": "Interval",
    "Oem": {
      "Dell": {
        "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
        "CustomLabel": null,
        "FQDD": null,
        "Source": null
      }
    }
  },
  {
    "MetricId": "IOUsage",
    "MetricProperties": [],
    "MetricProperties@odata.count": 0,
    "CollectionFunction": "Maximum",
    "CollectionDuration": "PT0H1M0S",
    "CollectionTimeScope": "Interval",
    "Oem": {
      "Dell": {
        "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
        "CustomLabel": null,
        "FQDD": null,
        "Source": null
      }
    }
  },
  {
    "MetricId": "IOUsage",

```

```

        "MetricProperties": [],
        "MetricProperties@odata.count": 0,
        "CollectionFunction": "Minimum",
        "CollectionDuration": "PT0H1M0S",
        "CollectionTimeScope": "Interval",
        "Oem": {
            "Dell": {
                "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
                "CustomLabel": null,
                "FQDD": null,
                "Source": null
            }
        },
        {
            "MetricId": "MemoryUsage",
            "MetricProperties": [],
            "MetricProperties@odata.count": 0,
            "CollectionFunction": "Average",
            "CollectionDuration": "PT0H1M0S",
            "CollectionTimeScope": "Interval",
            "Oem": {
                "Dell": {
                    "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
                    "CustomLabel": null,
                    "FQDD": null,
                    "Source": null
                }
            }
        },
        {
            "MetricId": "MemoryUsage",
            "MetricProperties": [],
            "MetricProperties@odata.count": 0,
            "CollectionFunction": "Maximum",
            "CollectionDuration": "PT0H1M0S",
            "CollectionTimeScope": "Interval",
            "Oem": {
                "Dell": {
                    "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
                    "CustomLabel": null,
                    "FQDD": null,
                    "Source": null
                }
            }
        },
        {
            "MetricId": "MemoryUsage",
            "MetricProperties": [],
            "MetricProperties@odata.count": 0,
            "CollectionFunction": "Minimum",
            "CollectionDuration": "PT0H1M0S",
            "CollectionTimeScope": "Interval",
            "Oem": {
                "Dell": {
                    "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
                    "CustomLabel": null,
                    "FQDD": null,
                    "Source": null
                }
            }
        },
        {
            "MetricId": "AggregateUsage",
            "MetricProperties": [],
            "MetricProperties@odata.count": 0,
            "CollectionFunction": "Average",
            "CollectionDuration": "PT0H1M0S",
            "CollectionTimeScope": "Interval",

```

```

        "Oem": {
            "Dell": {
                "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
                "CustomLabel": null,
                "FQDD": null,
                "Source": null
            }
        },
        {
            "MetricId": "AggregateUsage",
            "MetricProperties": [],
            "MetricProperties@odata.count": 0,
            "CollectionFunction": "Maximum",
            "CollectionDuration": "PT0H1M0S",
            "CollectionTimeScope": "Interval",
            "Oem": {
                "Dell": {
                    "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
                    "CustomLabel": null,
                    "FQDD": null,
                    "Source": null
                }
            },
            {
                "MetricId": "AggregateUsage",
                "MetricProperties": [],
                "MetricProperties@odata.count": 0,
                "CollectionFunction": "Minimum",
                "CollectionDuration": "PT0H1M0S",
                "CollectionTimeScope": "Interval",
                "Oem": {
                    "Dell": {
                        "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
                        "CustomLabel": null,
                        "FQDD": null,
                        "Source": null
                    }
                }
            },
            "Metrics@odata.count": 12,
            "Links": {
                "Triggers": [],
                "Triggers@odata.count": 0
            },
            "Oem": {
                "Dell": {
                    "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
                    "Digest": "bffb0c579c53d24c7bf3fb8fdaf6c0ff4ccd487072bab02c16c7864827abe420",
                    "iDRACFirmwareVersion": "5.10.00.00"
                }
            }
        }
    }
}

```

Metric Report

A Metric Report is a DMTF Telemetry specification-compliant JSON document that consists of metric names, metric values, and timestamps. The properties of a MetricReport are not configurable, they are read-only.

Topics:

- [Properties in a Metric Report](#)
- [List of Predefined Metric Reports Supported in Telemetry](#)

Properties in a Metric Report

The properties in a metric report are as follows:

- **Id** – The user-friendly name for this report. This generally matches the Id value for MetricReportDefinition, however, for MetricReportDefinition with ReportUpdates set to NewReport, this name has a dash and timestamp appended representing when the report was generated
- **Name** – Taken from the Name property in MetricReportDefinition
- **ReportSequence** – This is present for OnChange and Periodic report types only. This is a monotonically increasing index number that always starts at “1” when the report is enabled and increases by one with each report generated. Use case is for users to detect if they have missed collecting a report. This property is NOT PRESENT in OnRequest reports.
- **MetricReportDefinition/@odata.id** – Link to the Definition that controls this metric report
- **Timestamp** – The time that the report was generated. All MetricValue/timestamp(s) in the report will be less than or equal to this value
- **MetricValues** – An array of MetricValue(s)
 - **MetricId** – The name of the metric being reported.
 - **Timestamp** – Time in ISO format, UTC time (ends in “Z”) when the reading was taken.
 - **OEM/Dell/ContextID** – The intent of this field is to provide a uniform identifier for the name of the device that produced the metric. It is entered with the friendly FQDD for the device, however, some metrics are not connected to a device with an FQDD, so this is generated as a pseudo FQDD.
 - **OEM/Dell/Label** – The intent of this field is to provide a uniform or stable way for consumers to ingest and label metrics into their database backends. It consists of the ContextID and the Metric ID concatenated together. If any CollectionFunction(s) are applied, those are also appended to the name to disambiguate them.
 - **OEM/Dell/Source** – This field shows the data source name that the metric value was obtained from.
 - **OEM/Dell/FQDD** – The iDRAC standard FQDD of the device producing the metric.

List of Predefined Metric Reports Supported in Telemetry

Table 6. List of Predefined Metric Reports Supported in Telemetry

Metric Reports	Description	Hardware Peripherals Requirement	Notes
AggregationMetrics	The Aggregation Metrics Report combines base metrics for Power, Temperature, and CUPS (CPU, Memory, I/O, System) using formulas or filters to give a summarized view of system performance.	Common	By default, OnRequest ReportTimespan by default is 2 minutes. If the ReportTimespan is less than 2 minutes, you might get 0 odatacount intermittently.

Table 6. List of Predefined Metric Reports Supported in Telemetry (continued)

Metric Reports	Description	Hardware Peripherals Requirement	Notes
CPUMemMetrics	This Report offers key information about CPU and memory usage. It helps users track system performance, resource utilization, and identify potential bottlenecks.	Common	No specific Notes
CPURegisters	CPU Register dump (uuencoded binary data). This is platform-specific. On the Intel platform, this represents the MSR registers and on the AMD platform this represents the MCA registers.	Common	This report is applicable in 16G and previous generations. It is an MSR dump, this report is generated when an IERR event happens.
CPUSensor	CPU dynamic data includes CPU status, temperature, target power, and other metrics	Common	No specific Notes
FCPortStatistics	FC dynamic data includes port statistics and sensor (temperature and power) data	FC card	Metric values are reported at Port Level. For example, FC.Slot.1-1, FC.Slot.1-2
FCSensor	FC dynamic data includes sensor temperature data.	FC card	Metric values are reported at Slot Level, though the FQDD sees first Port on the slot For example, FC.Slot.1-1
FPGASensor	FPGA dynamic data includes temperature sensors	FPGA card	If FPGA is available in the system, a Telemetry report is generated. Note that FPGA is about to reach "End of Life" soon and this report will be deprecated.
FanSensor	Fan dynamic data includes fan health, PWM calculation, current speed (tach1 and tach2).	Common	Not supported on modular Platforms
GPUMetrics	Graphics Processing Unit Health metrics	NVIDIA GPU	Only metrics from NVIDIA GPUs
GPUStatistics	Graphics Processing Unit Frame Buffer and Graphic Device (GR) memory ECC(Error-correcting code) statistics data. This report is offered as an experimental preview.	NVIDIA GPU	Error Correction (Ecc) data are available only from faulty GPU cards.
GPUStatistics	Graphics Processing Unit Frame Buffer and Graphic Device (GR) memory ECC(Error-correcting code) statistics data. This report is offered as an experimental preview.	NVIDIA GPU	Error Correction (Ecc) data are available only from faulty GPU cards.
MemoryMetrics	The basic memory metrics for a memory device	Common	PredictedMediaLifeLeftPercent – Applicable to only

Table 6. List of Predefined Metric Reports Supported in Telemetry (continued)

Metric Reports	Description	Hardware Peripherals Requirement	Notes
			NVDIMM, rest of all metrics are applicable to DIMMs.
MemorySensor	Memory dynamic data includes CPU status, temperature, target power, and other metrics.	Common	No specific notes
NICSensor	NIC dynamic data includes port, partition and RDMA statistics, sensor (temperature and power) data	NIC/OCP Cards	No specific notes
NICStatistics	NIC dynamic data includes port, partition and RDMA statistics, sensor (temperature and power) data.	NIC/OCP cards	No specific notes
NVMeSMARTData	NVMe dynamic data includes NVMe SMART health data, and static data includes Inventory	NVMe Drives	NVMeSMARTData is only supported for SSD (PCIeSSD/ NVMe Express) drives with PCIe bus protocol (not behind SWRAID)
PSUMetrics	PSU dynamic data includes PSU health, output power, input, and output (volts, amps, watts).	Common	Only supported on Monolithic servers.
PowerMetrics	Power dynamic data includes power consumption for all CPUs, DIMMs, System Input, and System Output	Common	No specific notes
PowerStatistics	System power consumption statistics	Common	No specific notes
SFPTransceiver	Small Form Factor pluggable(SFP) optical xcvr data for NIC, FC, and infiBand	SFP Transceiver	SFP Transceiver device inserted in FC card
Sensor	All sensors (IPMI based)	Common	No specific notes
SerialLog	Serial log from host	Note: The Serial Communication option is applicable only if the serial COM port is installed in the system. Enable the Serial redirection on COM2 port. To enable the capturing of serial logs through racadm: racadm set iDRAC.SerialCapture.Enable enabled For more information, see Reference guide and iDRAC User guide	Deprecated from 17g and future versions.

Table 6. List of Predefined Metric Reports Supported in Telemetry (continued)

Metric Reports	Description	Hardware Peripherals Requirement	Notes
StorageDiskSMARTData	SSD SMART information	SAS/SATA Drives	StorageDiskSMARTData report is only supported for SSD drives behind PERC/HBA with SAS/SATA bus protocol.
StorageSensor	Temperature information for the internal storage drives	SAS/SATA or NVMe Drives	StorageSensor report is only supported for the drives in non-raid mode and not behind the BOSS controller.
SystemUsage	System Usage in percent. This report is platform-dependent, and the data may not be available on all platforms	Common	All metrics are supported on Intel, and only CPUUsage metric is supported on AMD.
ThermalMetrics	Thermal dynamic data includes device temperature and ambient temperature	Common	No notes specified
ThermalSensor	Thermal dynamic data includes device temperature and ambient temperature	Common	No notes specified
x86SubsystemPower	x86 Subsystem Power consumption for CPU, Memory, Storage, Fans, and PCIe	Common	Similar to PowerMetrics (six metrics from the PowerMetrics report added)

Metric Definition

Metric Definition tells us about metric. Each metric (Metric Definition) includes description, type, units, and sensing interval so forth, can be obtained using the following command.

To view all MetricDefinitions:

```
Command: GET
URI: https://<iDRAC_IP>/redfish/v1/TelemetryService/MetricDefinitions
Header: content-type application/json
```

For example:

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

To view a single MetricDefinition:

```
Command: GET/
URI: https://<iDRAC_IP>/redfish/v1/TelemetryService/MetricDefinitions<MetricID>
Header: content-type application/json
e.g.<MetricID> = BoardTemperature
```

For example:

```
curl -s -k -u <user>:<password> -X GET https://<IDRAC_IP>/redfish/v1/TelemetryService/MetricDefinitions/BoardTemperature
```

Topics:

- [Sample Metric Definition SystemMaxPowerConsumption](#)
- [Metrics](#)
- [System and Platform-Level Monitoring](#)

Sample Metric Definition 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": "Numeric",
  "Units": "W",
  "Accuracy": 1,
  "SensingInterval": "PT60S"
}
```

Metrics

This list provides detailed metric definitions that are related to various components monitored by iDRAC. Metric availability and definitions may change depending on the iDRAC version. To get the most accurate and updated list of metrics, users are advised to perform a Redfish pull on MD Collection and desired MD.

Network Devices

The provided text covers various topics that are related to network devices, including Fibre Channel Host Bus Adapters (FC HBAs), Network Interface Cards (NICs), and configuration options. It also touches on the importance of proper measurement units and formatting in technical writing, as outlined in the Dell Technologies Unified Style Guide. The text includes information about supported FC HBA devices, such as Emulex and QLogic, and their configuration options, as well as details on network devices like NICs, Converged Network Adapters (CNAs), and LAN On Motherboards (LOMs).

NIC Sensor

The NIC Sensor provides temperature readings in Celsius, with a sensing interval of 5 seconds, and is a numeric metric. NIC sensor monitors temperature and triggers alerts when thresholds are exceeded. Sensor readings detect potential issues.

Table 7. NIC Sensor

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
TemperatureReading	Temperature sensor reading in Celsius	Numeric	Celsius	5

NIC Statistics

NIC statistics provide metrics on packet transmission, reception, and error rates, with a sensing interval of 60 seconds, to monitor network performance.

Table 8. NIC Statistics

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
PartitionOSDriverState	Indicates partitions operating system driver states	Discrete (Unknown, Operational, Non-operational)	This is an enumeration of predefined states, so no units are required	60
TxMutlicast	Total number of good multicast packets transmitted in Pkt	Integer	Pkt	60
RxErrorPktAlignmentErrors	Total number of packets received with alignment errors in Pkt	Integer	Pkt	60
TxPauseXOFFFrames	Number of XOFF frames that are transmitted to the network	Integer	Pkt	60
OSDriverState	Indicates operating system driver states (Unknown, Operational, Non-operational)	Discrete(Unknown, Operational, Non-operational)	This is an enumeration of predefined states, so no units are required.	60

Table 8. NIC Statistics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
TxPauseXONFrames	Number of XON frames that are transmitted to the network in Pkt	Integer	Pkt	60
LinkStatus	Indicates whether the link is up or down	Discrete (Unknown, Up, Down)	This is an enumeration of predefined states, so no units are required	60
TxUnicast	Total number of good multicast packets transmitted in Pkt	Integer	Pkt	60
LanUnicastPktTxCount	Total number of Unicast LAN Packets Transmitted in Pkt	Integer	Pkt	60
PartitionLinkStatus	Indicates whether the partition link is up or down	Discrete (Unknown, Up, Down)	This is an enumeration of predefined states, so no units are required	60
TxErrorPktSingleCollision	Number of times that a successfully transmitted packet encountered a single collision	Counter	This is a count of collisions, so no units are required.	60
RDMA RxTotalPackets	Total number of RDMA packets received in Pkt	Counter	Pkt	60
TxErrorPktMultipleCollision	Number of times that a transmitted packet encountered more than one collision but fewer than 16	Counter	This is a count of multiple collisions, so no units are required.	60
RxBytes	Total number of bytes received, including host and remote management pass through traffic (remote management pass through traffic is applicable to LOMs only)	Counter	Bytes	60
RxPauseXONFrames	Flow control frames from the network to resume transmission (measured in pkts).	Counter	Pkt	60
RxErrorPktFCSErrors	Total number of packets received with Frame Check Sequence (FCS) errors	Counter	Pkt	60
RxBroadcast	Total number of good broadcast packets received	Counter	Pkt	60
RxRuntPkt	Total number of frames that are too short (< 64 bytes)	Counter	Pkt	60
RDMA TxTotalWritePkts	Total numbers of RDMA write packets	Counter	Pkt	60

Table 8. NIC Statistics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
	transmitted (measured in Pkt)			
RxUnicast	Total number of good unicast packets received (measured in Pkt)	Counter	Pkt	60
RDMATxTotalSendPkts	Total number of RDMA Send packets transmitted (measured in Pkt)	Counter	Pkt	60
TxBroadcast	Total number of good broadcast packets transmitted. (measured in Pkt)	Counter	Pkt	60
RDMARxTotalBytes	Total number of RDMA bytes received	Counter	Bytes	60
RDMATxTotalReadReq Pkts	Total number of RDMA Read Request packets transmitted	Counter	Pkt	60
RDMATxTotalPackets	Total number of RDMA packets transmitted	Counter	Pkt	60
TxErrorPktExcessiveCollision	Number of times that encountered 16 or more collisions that occurred on a single transmit packet	Counter	No units required	60
RDMATxTotalBytes	Total number of RDMA bytes transmitted	Counter	Bytes	60
RDMATotalProtocolErrors	Total number of RDMA Protocol errors	Counter	No units required	60
TxErrorPktLateCollision	Number of collisions that occurred after one slot time	Counter	No units required	60
RxFalseCarrierDetection	Total number of false carrier errors received from Physical Layer Device (PHY).	Counter	Pkt	60
LanUnicastPktRxCount	Total number of LAN Unicast Packets Received	Counter	Pkt	60
RxJabberPkt	Total number of incoming network frames that are larger than the allowed maximum frame size.	Counter	Pkt	60
RDMATotalProtectionErrors	Total number of RDMA Protection errors	Counter	No units required	60
TxBytes	Total number of bytes transmitted, including host and	Counter	Bytes	60

Table 8. NIC Statistics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
	remote management pass through traffic			
LanFCSRxErros	Number of Rx packets with Checksum errors. (measured in Pkt)	Counter	Pkt	60
RxMulticast	Total number of good multicast packets transmitted	Counter	Pkt	60
RxPauseXOFFFrames	The number of pause frames received from the network, instructing the server to temporarily stop sending data to help manage network congestion. (measured in Pkt)	Counter	Pkt	60
DiscardedPkts	Total number of discarded packets	Counter	Pkt	60
FCOERxPktDroppedCount	Number of packets received with FCS errors	Counter	No units required	60
FCCRCErrorsCount	Number of FC frames with CRC errors	Counter	Pkt	60
FCOELinkFailures	Number of FCoE/FIP Login failures	Counter	No units Required	60
FCOEPktRxCount	Number of good (FCS valid) packets received with the partition active Fiber Channel over Ethernet (FCoE) MAC address.	Counter	Pkt	60
FCOEPktTxCount	Number of good (FCS valid) packets transmitted that passed L2 filtering by a specific MAC address. (measured in Pkt)	Counter	Pkt	60

Fiber Channel Port Statistics (FCPort)

Fiber Channel Port Statistics displays data transfer rates, errors, and performance metrics for Fiber Channel ports, aiding in storage network troubleshooting and optimization efforts quickly.

Table 9. Fiber Channel Port Statistics (FCPort)

Metric Ids	Description	Metric Type	Units	Sensing Interval (seconds)
FCRxKBCount	This property represents the total KBs received	Counter	KB	60

Table 9. Fiber Channel Port Statistics (FCPort) (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval (seconds)
FCRxSequences	This property represents the FC sequences received	Counter	This is a count of received sequences, so no units are required	60
FCLinkFailures	This property represents the number of link failures	Counter	This is a count of link failures, so no units are required	60
FCInvalidCRCs	This property represents the count of invalid CRCs	Counter	This is a count of invalid CRCs, so no units are required	60
PortSpeed	Specifies the port speed	Discrete (No Link, 2 Gbps, 4 Gbps, 8 Gbps, 16 Gbps, 32 Gbps, 64 Gbps, 128 Gbps, 256 Gbps, Unknown)	This is an enumeration of predefined port speeds, so no units are required.)	60
FCRxTotalFrames	Specifies the total FC frames received	Counter	Pkt	60
PortStatus	FC port status	Discrete(Down, Up, Unknown)	No units (because it is a status, not a numeric measurement)	60
FCStatOSDriverState	This property indicates the operating system driver state	Discrete (Other, Not Applicable, Non-operational, Operational, Unknown)	No units (because it is a status, not a numeric measurement)	60
FCTxKBCount	This property represents the total KBs transmitted	Counter	KB	60
FCLossOfSignals	This property represents the lost signals count	Counter	This is a count of lost signals, so no units are required.	60
FCTxSequences	This property represents the FC sequences transmitted	Counter	This is a count of transmitted sequences, so no units are required	60
FCTxTotalFrames	Specifies the total FC frames transmitted	Counter	This is a count of transmitted frames, so no units are required	60

Fiber Channel Sensors

The Fiber Channel Sensors provide temperature readings in Celsius, with a sensing interval of 5 s, and the metric type is numeric.

Table 10. Fiber Channel Sensors

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
TemperatureReading	Temperature sensor reading in Celsius	Numeric	Celsius	5

SFP Transceiver Metrics

The SFP Transceiver Metrics provide various measurements for Small Form Factor pluggable (SFP) transceivers, including Rx input power, operating voltage, temperature, Tx bias current, and Tx output power, with each metric having a specific unit and sensing interval of 60 s. These metrics are categorized into numeric and discrete types, with discrete metrics indicating status such as Unknown, OK, Warning, or Critical. The metrics are crucial for monitoring and maintaining the health and performance of SFP transceivers in various applications.

Table 11. SFP Transceiver Metrics

Metric Ids	Description	Metric Type	Units	Sensing Interval (seconds)
RxInputPower	Rx Input power value of Small Form Factor pluggable (SFP) in milliwatts	Numeric	milliWatts	60
SFPVoltage	Measures the operating voltage of the SFP transceiver sensor, reported in volts (V).	Numeric	Voltage	60
TemperatureStatus	The SFP transceiver's temperature sensor.	Discrete (Unknown, OK, Warning, Critical)	For Temperature Status, no units needed—it is a status indicator, not a numeric reading.	60
TxBiasCurrent	Tx Bias current value of Small Form Factor pluggable(SFP) in milliamps	Numeric	milliAmps	60
TxBiasCurrentStatus	Status of Tx Bias Current value of Small Form Factor pluggable(SFP) Transceiver Metric Definition	Discrete (Unknown, OK, Warning, Critical)	For TxBiasCurrentStatus, there are no units—because it is a status indicator, not a direct numeric measurement.	60
TxOutputPower	Tx output power value of Small Form Factor pluggable(SFP) in Watts	Numeric	milliWatts	60
TxOutputPowerStatus	Indicates the status of Tx Output Power value limits of Small Form Factor pluggable(SFP) Transceiver	Discrete(Unknown, OK, Warning, Critical)	TxOutputPowerStatus is a status indicator, not a direct measurement. Therefore, it does not have units.	60
VoltageStatus	Status of the Voltage of Small Form Factor pluggable(SFP) Transceiver	Discrete (Unknown, OK, Warning, Critical)	For SFP Voltage Status, there is no unit as it is status	60
SFPTemperature	Status of Temperature sensor reading of Sfp transceiver in Celsius	Numeric	Celsius	60
RxInputPowerStatus	Indicates the status of Rx Input Power value limits of Small Form Factor pluggable(SFP) Transceiver	Discrete (Unknown, OK, Warning, Critical)	Not Applicable	60

DPU/SmartNiCSensor

The DPU/SmartNiCSensor metrics include DPU Temperature and DPU Power Consumption, which are reported in Celsius and Watts, respectively, with a sensing interval of 5 s. The metrics are used to monitor the temperature and power consumption of the DPU, providing valuable insights for system maintenance and optimization.

Table 12. DPU/SmartNiCSensor

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
DPUTemperature	DPU Temperature sensor reading in Celsius	Numeric	Celsius	5
DPUPowerConsumption	DPU Power Consumption in Watts	Numeric	Watts	5

CPU and Memory Metrics

The following discusses various aspects of Dell server metrics, including CPU and memory metrics, system usage reports, and telemetry performance reports. It outlines the different types of metrics available, such as CPU usage, memory usage, and power consumption, and explains how to access and analyze these metrics using tools like OpenManage Enterprise. The text also touches on the importance of monitoring server performance and identifying areas for improvement to ensure efficient operation.

CPU Registers

CPU registers provide low-level insights into processor operations, such as instruction execution and performance counters, with platform-specific representations like MSR registers on Intel platforms and MCA registers on AMD platforms. The Compute Usage Per Second (CUPS) functionality, supported by Intel ME, offers real-time monitoring of CPU, memory, and I/O utilization. CPU utilization is based on time that is spent in active and inactive states, with data from Resource Monitoring Counters (RMCs) aggregated to measure cumulative utilization.

CPU Sensors

The CPU Sensors provide temperature readings in Celsius, with a sensing interval of 5 s, and the temperature reading is a numeric value. The CPU temperature sensor reading can be monitored using the Power Manager, and the thermal metrics can be viewed on the server details page. The temperature reading is an important metric for monitoring CPU performance and preventing overheating.

Table 13. CPU Sensors

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
TemperatureReading	CPU Temperature sensor reading in Celsius	Numeric	Celsius	5

CPU Metrics

The provided text lists various CPU metrics, including frequency, energy, and power, along with their descriptions, metric types, and units. These metrics, such as AvgFrequencyAcrossCores, DRAMPkgEnergy, and CPUPkgEnergy, are used to measure CPU performance and power consumption. The metrics are categorized by type, including numeric, integer, and counter, and have varying sensing intervals and units of measurement.

Table 14. CPU Metrics

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
AvgFrequencyAcrossCores	Average CPU frequency across all the cores. (measured in MHz)	Numeric	Mega-Hertz (MHz)	5
DDRLimitingCounter	Accumulated DRAM throttle time. Read the sum of all-time durations for which each DIMM has been throttled. (measured in milliseconds)	Numeric	milliseconds (mS)	5
DRAMPkgEnergy	Accumulated DRAM energy consumed by all the DIMMs in all the channels. (measured in micro Joule)	Numeric	micro Joule (uJ)	5
DRAMPwr	DRAM power used by all the DIMMs on all the channels. (measured in Watts)	Numeric	Watts (W)	5
DRAMThrottling	Read the sum of all-time durations for which each DIMM has been throttled. (measured in mS)	Integer	milliseconds (mS)	5
EnergyTimestamp	Timestamp reported with the most recently acquired Energy reading as taken from the UNCORE Time Stamp counter. (measured in msec)	Numeric	milliseconds (msec)	5
LimitingEvents	Provides indications of reasons for performance being limited, 0 - IA32_PERF_LIMIT_REASONS; 1 - IA32_CLM_PERF_LIMIT_REASONS	Numeric	Not Applicable	5
NonC0ResidencyLow	32 LSBs of the 64-bit counter containing a number of Package C2 and C6 cycles	Counter	Not Applicable	5
NonC0ResidencyHigh	32 MSBs of the 64-bit counter containing a number of Package C2 and C6 cycles	Counter	Not Applicable	5
PkgThermalStatus	The register includes status and log bits for TCC activation PROCHOT_N assertion and Critical Temperature.	Numeric	Not Applicable	5

Table 14. CPU Metrics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
TCtrl	Value that indicates the relative offset below Prochot temperature at which fans should be engaged. (measured in Cel)	Numeric	Celsius (Cel)	5
TJMax	Prochot temperature where the Thermal Monitor is activated. Alternatively named Throttle Temperature or maximum junction temperature.	Numeric	Celsius (Cel)	5
ThermalCrtlCircuitActivation	Read the time for which the processor has been operating in a lowered power state due to internal TCC activation. (measured in milliseconds)	Numeric	milliseconds (mS)	5
UncoreClocksLow	32 LSBs of the 64-bit counter with Uncore (Mesh) clocks.	Numeric	Not Applicable	5
UncoreClocksHigh	32 MSBs of the 64-bit counter with Uncore (Mesh) clocks.	Numeric	Not Applicable	5
PkgPwr	CPU package power.	Numeric	Watts (W)	5
AccCoreCyclesLow	32 LSBs of the 64-bit counter containing accumulated Cx cycles of all enabled cores.	Numeric	Not Applicable	5
AccCoreCyclesHigh	32 MSBs of the 64-bit counter containing accumulated Cx cycles of all enabled cores.	Numeric	Not Applicable	5
CPUC0ResidencyLow	32 LSBs of the 64-bit counter containing the aggregate C0 residency count from all the cores	Counter	Not Applicable	5
CPUC0ResidencyHigh	32 MSBs of the 64-bit counter containing the aggregate C0 residency count from all the cores	Counter	Not Applicable	5
CPUEpi	Provides an approximation of the total instructions that are retired per time unit (that is not idle or not stalled for memory and so on). It is used by the platform to distribute power across the sockets.	Numeric	milliseconds (mS)	5

Table 14. CPU Metrics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
CPUAvgPbmRatioCounterLow	CPU Average PBM ratio counter	Numeric	Not Applicable	5
CPULimitingCounter	The duration for which the RAPL power limit determined the socket operating condition.	Numeric	milliseconds (mS)	5
CPUpkgEnergy	Accumulated energy consumed by the CPU Package	Numeric	micro Joule (uJ)	5
CPUViolationCounter	Accumulated CPU throttle time. Duration for which the processor was throttled below PERF_CTL (Host operating system) request due to power limit.	Numeric	milliseconds (mS)	5
CUPSIOBandwidthDMI	Utilization counter of the DMI Port	Numeric	Percentage (%)	5
CUPSIOBandwidthPort 0	Utilization counter of the integrated I/O Port on a per x16 port granularity	Numeric	Percentage (%)	5
CUPSIOBandwidthPort 1	Utilization counter of the integrated I/O Port on a per x16 port granularity	Numeric	Percentage (%)	5
CUPSIOBandwidthPort 2	Utilization counter of the integrated I/O Port on a per x16 port granularity	Numeric	Percentage (%)	5
CUPSIOBandwidthPort 3	Utilization counter of the integrated I/O Port on a per x16 port granularity	Numeric	Percentage (%)	5

Memory Sensor

The Memory Sensor reports temperature readings in Celsius, with a sensing interval of 5 s, and provides information about DIMM temperature.

Table 15. Memory Sensor

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
TemperatureReading	Temperature sensor reading in Celsius	Numeric	Celsius	5

Cumulative System Usage

The provided text describes various cumulative system usage metrics, including CPU, memory, and I/O usage, which are measured as percentage readings and provide insights into system utilization and resource consumption. These metrics, such

as CPUUsage, MemoryUsage, and IOUsage, are collected at regular intervals, typically 5 s, and offer a comprehensive view of system performance. The metrics are categorized by type, units, and sensing interval, allowing for detailed analysis and monitoring of system usage.

Table 16. Cumulative System Usage

Metric Ids	Description	Metric Type	Units	Sensing Interval (seconds)
CPUUsage	The combined CPU utilization of all the CPU cores in the system as a percentage reading. Utilization is based on time that is spent in an active state compared to time spent in an inactive state.	Numeric	Percentage (%)	5
MemoryUsage	There are individual counters to measure memory traffic occurring at each memory channel or memory controller instance. These counters are combined to measure the cumulative memory traffic across all the memory channels on the system. This number is a measure of memory bandwidth consumption and not the capacity of the system memory.	Numeric	Percentage (%)	5
IOUsage	Each root port in the PCIe root complex has a counter that measures the traffic from or directed to the root port. These counters are combined to measure traffic for all PCIe segments derived from the package. This aggregate number is a measure of PCIe root complex I/O bandwidth utilization for the system.	Numeric	Percentage (%)	5
AggregateUsage	This value is calculated by combining CPU, memory, and I/O indexes considering the load factor of each resource. The load factor depends on the nature of the workload on the system.	Numeric	Percentage (%)	5
CPUMaxUsage	CPU Max Usage Percent	Numeric	Percentage (%)	5

Table 16. Cumulative System Usage (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval (seconds)
CPUUsageHighFrequency	CPU Usage Percent sampled at higher frequency, that is, 200 ms	Numeric	Percentage (%)	200
MemoryMaxUsage	Memory Max Usage Percent	Numeric	Percentage (%)	5
IOMaxUsage	I/O Max Usage Percent	Numeric	Percentage (%)	5
AggregateMaxUsage	CPU Max Usage Percent	Numeric	Percentage (%)	5
CPUUsagePctReading	CPU system utilization as percentage reading. Data from RMCs (dedicated chipset Resource Monitoring Counters) for each CPU core is aggregated to provide cumulative utilization of all the cores in the system. This utilization is based on time that is spent in active and inactive states.	Numeric	Percentage (%)	5
MemoryUsagePctReading	Utilization of memory bandwidth represented in percentage terms. RMCs (dedicated chipset Resource Monitoring Counter) measure memory traffic occurring at each memory channel or memory controller instance. Data from these RMCs is aggregated to measure the cumulative memory traffic across all the memory channels on the system. This measure can indicate if the workload is memory intensive or not.	Numeric	Percentage (%)	5
IOUsagePctReading	I/O bandwidth utilization for the system as a percentage reading. There is one RMC (dedicated chipset Resource Monitoring Counter) per root port in the PCI Express Root Complex to measure PCI Express traffic emanating from or directed to that root port and the	Numeric	Percentage (%)	5

Table 16. Cumulative System Usage (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval (seconds)
	lower segment. Data from these RMCs is aggregated for measuring PCI express traffic for all PCI Express segments emanating from the package.			
SystemUsagePctReading	System utilization aggregates CPU, Memory, and I/O index considering a predefined load factor of each system resource. The load factor depends on the nature of the workload on the system. This represents the measurement of the compute headroom available on the server.	Numeric	Percentage (%)	5

Receiving Telemetry Reports

After Telemetry streaming is configured on the iDRAC, Redfish clients can either stream Telemetry reports continuously or pull Telemetry reports on demand. The following sections describe the methods through which clients can receive the data.

General Sensor Metrics

The provided text lists various sensor metrics, including temperature, voltage, and current readings, along with their corresponding metric IDs, descriptions, and sensing intervals. The metrics are categorized by sensor type, such as CPU, memory, and fan sensors. The text also provides information about telemetry performance reports, which include sensor data indicating compute usage, power consumption, and aggregate temperature readings.

Table 17. General Sensor Metrics

Metric Ids	Description	Metric Type	Sensing Interval (seconds)
MemoryUsagePctReading	The reading of the Sensor	Numeric	5
RPMReading	Current reading specific to the sensor	Numeric	5
AmpsReading	The reading of the Sensor	Numeric	5
TemperatureReading	Temperature sensor reading in Celsius	Numeric	5
CPUUsagePctReading	The reading of the Sensor	Numeric	5
IOUsagePctReading	The reading of the Sensor	Numeric	5
VoltageReading	The reading of the Sensor	Numeric	5
WattsReading	The reading of the Sensor	Numeric	5
SystemUsagePctReading	The reading of the Sensor	Numeric	5

Chassis Level and Environmental Metrics

Server environmental metrics encompass a comprehensive array of telemetry data that are crucial for monitoring and optimizing power, thermal, and system usage characteristics to maintain peak server performance and efficiency. These metrics are essential for ensuring the reliable operation of servers by providing deep insights into power consumption, thermal conditions, and overall system health. The data is organized into key categories, including Power Metrics, Power Statistics, x86 Subsystem Power Metrics, Thermal Metrics, Thermal Sensors, System Usage, and System Aggregation Metrics. Each category delivers both real-time and historical data on power efficiency, thermal management, system utilization, and environmental factors.

Power Metrics (Server Motherboard)

The provided text lists various power metrics for server motherboards, including CPU power consumption, memory power consumption, and fan power consumption, among others. These metrics are used to measure and monitor the power usage of different server components. The text also provides information about the units of measurement, sensing intervals, and metric descriptions for each power metric.

Table 18. Power Metrics (Server Motherboard)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
Metric Ids	Description	Metric Type	Units	Sensing Interval (seconds)
CPUPower	Per socket CPU power consumption in W.	Numeric	Watts (W)	5
FPGAPower	Per FPGA Sensor Power Consumption in W.	Numeric	Watts (W)	5
TotalCPUPower	Overall CPU power consumption in W.	Numeric	Watts (W)	5
TotalFPGAPower	Overall FPGA power consumption in W.	Numeric	Watts (W)	5
TotalPciePower	Overall PCIe power consumption in W.	Numeric	Watts (W)	5
TotalMemoryPower	Overall memory power consumption in W.	Numeric	Watts (W)	5
TotalFanPower	Overall Fan power consumption in W.	Numeric	Watts (W)	5
TotalStoragePower	Overall Storage power consumption in W.	Numeric	Watts (W)	5
SystemInputPower	Current System Input Power in W.	Numeric	Watts (W)	5
SystemOutputPower	Current System Output Power in W.	Numeric	Watts (W)	5
SystemHeadRoomInstantaneous	Instantaneous Headroom in W.	Numeric	Watts (W)	5
SystemPowerConsumption	Present reading in watts in W.	Numeric	Watts (W)	5

Power Statistics

The provided text describes various power statistics metrics, including maximum, average, and minimum power consumption over different time intervals such as last day, hour, minute, and week, with units measured in watts and sensing intervals of 60

s. These metrics are further categorized by their type, with some being numeric and others being datetime strings. The metrics are used to track and analyze power consumption patterns over various time periods.

Table 19. Power Statistics

Metric Ids	Description	Metric Type	Units	Sensing Interval (seconds)
LastDayMaxPower	Last Day Max Power	Numeric	Watts (W)	60
LastDayAvgPower	Last Day Average Power	Numeric	Watts (W)	60
LastDayMinPower	Last Day Min Power	Numeric	Watts (W)	60
LastHourMaxPower	Last Hour Max Power	Numeric	Watts (W)	60
LastHourAvgPower	Last Hour Average Power	Numeric	Watts (W)	60
LastHourMinPower	Last Hour Min Power	Numeric	Watts (W)	60
LastMinuteMaxPower	Last Minute Max Power	Numeric	Watts (W)	60
LastMinuteAvgPower	Last-Minute Average Power	Numeric	Watts (W)	60
LastMinuteMinPower	Last Minute Min Power	Numeric	Watts (W)	60
LastWeekMaxPower	Last Week Max Power	Numeric	Watts (W)	60
LastWeekAvgPower	Last Week Average Power	Numeric	Watts (W)	60
LastWeekMinPower	Last Week Min Power	Numeric	Watts (W)	60
LastWeekMaxPowerTime	Last Week Max Power Time	String	Not applicable as its metric datatype is datetime	0
LastWeekMinPowerTime	Last Week Min Power Time	String	Not applicable as its metric datatype is datetime	0
LastDayMaxPowerTime	Last Day Max Power Time	String	Not applicable as its metric datatype is datetime	0
LastDayMinPowerTime	Last Day Min Power Time	String	Not applicable as its metric datatype is datetime	0
LastHourMaxPowerTime	Last Hour Max Power Time	String	Not applicable as its metric datatype is datetime	0
LastHourMinPowerTime	Last Hour Min Power Time	String	Not applicable as its metric datatype is datetime	0
LastMinuteMaxPowerTime	Last Minute Max Power Time	String	Not applicable as its metric datatype is datetime	0
LastMinuteMinPowerTime	Last Minute Min Power Time	String	Not applicable as its metric datatype is datetime	0

x86 Subsystem Power Metrics

Tracks power consumption across CPU sockets, memory, fans, PCIe, and storage components to monitor and optimize x86 subsystem energy usage.

Table 20. x86 Subsystem Power Metrics

Metric Ids	Description	Metric Type	Units	Sensing Interval (seconds)
CPUPower	Per-socket CPU power consumption	Numeric	Watts (W)	5
TotalCPUPower	Overall CPU power consumption	Numeric	Watts (W)	5
TotalFanPower	Overall Fan power consumption	Numeric	Watts (W)	5
TotalMemoryPower	Overall memory power consumption	Numeric	Watts (W)	5
TotalPciePower	Overall PCIe power consumption	Numeric	Watts (W)	5
TotalStoragePower	Overall Storage power consumption	Numeric	Watts (W)	5

Thermal Metrics

Thermal metrics track PSU heat dissipation, airflow efficiency, temperature deltas, and power-to-cooling ratios to optimize thermal performance and energy efficiency in server environments.

Table 21. Thermal Metrics

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
TotalPSUHeatDissipation	Heat that is dissipated from the PSU as a result of its inefficiency	Numeric	British Thermal Units (BTU)	5
PSUEfficiency	PSU efficiency in percentage	Numeric	Percentage (%)	5
PowerToCoolRatio	Fan Power over system power ratio (PCTR)	Numeric	Not applicable	5
ComputePower	System power that is not wasted	Numeric	Watts (W)	5
SysRackTempDelta	Temperature difference between exhaust air and inlet air temperature of the server	Numeric	Celsius (Cel)	5
SysNetAirflow	System net airflow (measured in CFM)	Numeric	Cubic Feet per Minute (CFM)	5
SysAirflowUtilization	Specifies the system airflow utilization (CFM/Max CFM). CFM (cubic feet per min)	Numeric	Cubic Feet per Minute (CFM)	5
SysAirFlowEfficiency	System airflow efficiency metric (SAFE) in %.	Numeric	Percentage (%)	5

Table 21. Thermal Metrics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
SysAirflowPerSysInputPower	System airflow per unit of system input [DC] power consumption	Numeric	Cubic Feet per Minute (CFM)	5
SysAirflowPerFanPower	System airflow per unit of system fan power consumption	Numeric	Cubic Feet per Minute (CFM)	5
ITUE	Indicates the power utilization efficiency. It is expressed as the ratio of total system power to compute power	Numeric	Not Applicable	5

Thermal Sensor

Reports real-time GPU temperature in Celsius for thermal monitoring and system protection.

Table 22. Thermal Sensor

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
TemperatureReading	Temperature sensor reading in Celsius	Numeric	Celsius (Cel)	5

Voltage or Power or Amperage Sensor

Monitors real-time voltage, power, and current readings from sensors to assess electrical performance and ensure system stability.

Table 23. Voltage or Power or Amperage Sensor

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
VoltageReading	Voltage sensor reading in Volts.	Numeric	Voltage (V)	5
WattsReading	Power sensor reading in Watts.	Numeric	Watts (W)	5
AmpsReading	Amperage sensor reading in Amps.	Numeric	Amps (A)	5

Inlet Temperature and Power Aggregation Metrics

Tracks average and peak inlet temperature, peak power consumption, and cumulative energy usage to support thermal and power efficiency analysis.

Table 24. Inlet Temperature and Power Aggregation Metrics

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
SystemAvgInletTempHour	Average Inlet temperature for the last hour	Numeric	Celsius (Cel)	60

Table 24. Inlet Temperature and Power Aggregation Metrics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
SystemMaxInletTempHour	Peak Inlet temperature for the last hour	Numeric	Celsius (Cel)	60
SystemMaxPowerConsumption	Peak system power consumption	Numeric	Watts (W)	60
CumulativeSystemEnergy	Total power in energy consumed by server from start time	Numeric	Kilowatts-hour (KWH)	60

Fan Sensor

Reports current fan speed in RPM to monitor cooling performance and maintain optimal thermal conditions.

Table 25. Fan Sensor

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
RPMReading	Current reading specific to the sensor	Numeric	Celsius (Cel)	5

PSU Metrics

Monitors PSU temperature and fan speed to assess power supply health and cooling efficiency.

Table 26. PSU Metrics

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
PSUTemperatureReading	PSU Temperature sensor reading in Celsius	Numeric	Celsius (Cel)	60
PSURPMReading	PSU Fan reading in RPMs	Numeric	Revolutions Per Minute (RPM)	60

Accelerators

Accelerator metrics deliver critical data for optimizing the performance and reliability of GPUs and FPGAs. Monitoring indicators such as power consumption, memory usage, clock frequency, and error rates enables identification of system behavior patterns, detection of potential issues, and implementation of performance tuning strategies. This data supports improved efficiency, reduced downtime, and stable operation of hardware components in high-demand server environments.

GPU Metrics

GPU metrics offer comprehensive insights into the performance, health, and operational status of GPU components in server environments. These metrics include temperature readings, power consumption, memory usage, clock frequencies, PCIe throughput, NVLink activity, and various alert states. Each metric is captured at regular intervals and categorized as either numeric, discrete, or counter types, enabling precise monitoring and diagnostics for optimizing GPU reliability and efficiency.

Table 27. GPU Metrics

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
GPUStatus	GPU Availability Status	Discrete (Available, Unavailable, Not Applicable)	Not Applicable	5
MemoryTemperature	Temperature (degrees C) on GPU memory, if supported	Numeric	Celsius (Cel)	5
GPUHealth	GPU health status	Discrete (Unknown, OK, Degraded, Critical)	Not Applicable	5
PowerSupplyStatus	Specifies Power supply status	Discrete (Disabled, Enabled, Not Applicable)	Not Applicable	5
BoardPowerSupplyStatus	GPU board power supply status	Discrete (UnderPowered, SufficientPower, Not Applicable)	Not Applicable	5
BoardTemperature	Temperature (degrees C) on GPU board, if supported	Numeric	Celsius (Cel)	5
PowerBrakeState	Power brake state	Discrete (Released, Set, Not Applicable)	Not Applicable	5
PowerConsumption	Total GPU board power consumption in mWatts (100 mW resolution)	Numeric	milliwatts (mW)	5
ThermalAlertState	Thermal alert state	Discrete (NotPending, Pending, Not Applicable)	Not Applicable	5
PrimaryTemperature	Temperature (degrees C) on primary GPU	Numeric	Celsius (Cel)	5
SecondaryTemperature	Temperature (degrees C) on secondary GPU in dual GPU boards	Numeric	Celsius (Cel)	5
GPUMemoryClockFrequency	GPU Memory Clock Frequency	Numeric	Mega-Hertz (MHz)	5
GPUClockFrequency	GPU Clock Frequency	Numeric	Mega-Hertz (MHz)	5
GPUMemoryUsage	GPU Memory Usage Percent	Numeric	Percentage (%)	5
GPUUsage	GPU Usage Percent	Numeric	Percentage (%)	5
GPUEnforcedPowerLimit	GPU Configured Power Cap Limit (measured in mW)	Numeric	milliWatts (mW)	5
GPUArbitratedPowerLimit	GPU Current Power Cap Limit (measured in mW)	Numeric	milliWatts (mW)	5
GPUSMOccupancy	The ratio of number of warps residents on an SM to the theoretical maximum number of warps per elapsed cycle as a percentage	Numeric	Percentage (%)	5

Table 27. GPU Metrics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
GPUPCIELinkSpeedMax	Max PCIe Link Speed. 0 = Unknown, 1 = 2500 MTPS (PCIe Gen1.0), 2 = 5000 MTPS (PCIe Gen2.0), 3 = 8000 MTPS (PCIe Gen3.0), 4 = 16000 MTPS (PCIe Gen4.0), 5 = 32000 MTPS (PCIe Gen5.0), ...	Numeric	Not Applicable	5
GPUHmmaUsage	The ratio of cycles the tensor (HMMA) pipe is active (off the peak sustained elapsed cycles) as a percentage	Numeric	Percentage (%)	5
GPUMemBandwidthUsage	The memory Bandwidth utilization. DRAM usage as a percentage (relative to maximum theoretical bandwidth).	Numeric	Percentage (%)	5
GPUSMActivity	The ratio of cycles an SM has at least one warp assigned (computed from the number of cycles and elapsed cycles).	Numeric	Percentage (%)	5
GPUTotalSMUsageTime	Accumulated SM utilization time in millisecond. Clears on SBR or fundamental reset; no rollover protection	Counter	milliseconds (mS)	5
GPUTensorCoreUsage	The ratio of cycles any tensor pipe is active as a percentage.	Numeric	Percentage (%)	5
GPUPCIETxThroughput	PCIe TX throughput.	Numeric	Gbps	5
GPUPCIERxThroughput	PCIe RX throughput.	Numeric	Gbps	5
GPUClockEventReason	List of factors that are reducing clock frequency. Each bit represents one factor. The current defined bits (in order of bits from right to left) are - 1- ClockOptimizedForPower, 2-HWSlowdown, 3-HWThermalSlowdown, 4- HWPowBrakeSlowdown, 5-SyncBoost, and 6- ClockOptimizedForTher	Numeric	Not Applicable	5

Table 27. GPU Metrics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
	mal. Other bits might be defined in the future			
GPUPCIELinkSpeed	Current PCIe Link Speed. 0 = Unknown, 1 = 2500 MTPS (PCIe Gen1.0), 2 = 5000 MTPS (PCIe Gen2.0), 3 = 8000 MTPS (PCIe Gen3.0), 4 = 16000 MTPS (PCIe Gen4.0), 5 = 32000 MTPS (PCIe Gen5.0), ...	Numeric	Not Applicable	5
GPUPCIECorrectableErrorCount	Correctable error count of the following errors - Receiver error, - Bad TLP, - Bad DLLP, - Replay Rollover, - Replay timeout, - Advisory nonfatal error	Numeric	Not Applicable	5
GPUResetRecommendedState	A flag that indicates if a GPU reset is recommended. 0 - reset not recommended, 1 - reset recommended	Numeric	Not Applicable	5
GPUNVLinksCount	Count of NVLinks for the current device. Useful for other GPUNVLink metrics.	Numeric	Not Applicable	5
GPUNVLinkStatusFlag	Each bit represents a status flag for NVLink (0 - NVLink is down and 1 - NVLink is up). GPUNVLinksCount gives the number of bits to be read.	Numeric	Not Applicable	5
GPUNVLinkRuntimeError	Each bit represents a runtime error for NVLink (0 - no error occurred and 1 - error occurred). GPUNVLinksCount gives the number of bits to be read.	Numeric	Not Applicable	5
GPUNVLinkTrainingError	Each bit represents a training error for NVLink (0 - no error occurred and 1 - error occurred). GPUNVLinksCount gives the number of bits to be read.	Numeric	Not Applicable	5
GPUNVLinkRxThroughput	NVLink Rx data throughput (measured in Gbps)	Numeric	Gbps	1

Table 27. GPU Metrics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
GPUNVLinkTxThroughput	NVLink Tx data throughput (measured in Gbps)	Numeric	Gbps	1
GPUSWViolationDuration	Total duration of throttling (in nanoseconds) caused by global software violation of the processor since reset.	Numeric	nanoseconds (nSec)	1
GPUPowerViolationDuration	Total duration of throttling (in nanoseconds) caused by a power limit of the processor since a reset.	Numeric	nanoseconds (nSec)	1
GPUSWThermalViolationDuration	Total duration of throttling (in nanoseconds) caused by a thermal limit of the processor since reset.	Numeric	nanoseconds (nSec)	1
GPUDMMAUsage	The ratio of cycles the tensor (DFMA) pipe is active (off the peak sustained elapsed cycles)	Numeric	Percentage (%)	5

GPU Statistics

GPU statistics provide detailed error tracking and memory health insights for GPU components. These metrics focus on single-bit and double-bit error counters across various GPU memory regions, including frame buffer, L1/L2 cache, register files, and texture units. The data includes both real-time and cumulative error counts, and the number of memory pages retired due to errors. These statistics are essential for diagnosing hardware reliability, detecting memory faults, and maintaining GPU stability in high-performance computing environments.

Table 28. GPU Statistics

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
SBECOUNTERFB	Single-bit errors from Frame Buffer in Frame Buffer memory	Counter	Not Applicable	600
SBECOUNTERFBL2Cache	No of Single-bit errors from L2 cache in Frame Buffer memory	Counter	Not Applicable	600
SBECOUNTERGRRF	Single-bit errors from SM register file (RF) in Graphic Device (GR) memory	Counter	Not Applicable	600
SBERetiredPages	Number of pages dynamically retired because of single-bit errors in Frame Buffer memory	Counter	Not Applicable	600

Table 28. GPU Statistics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
SBECOUNTERGRTex	Single-bit errors from texture units (TEX) in Graphic Device (GR) memory	Counter	Not Applicable	600
SBECOUNTERGRL1Cache	Single-bit errors from L1 cache in Graphic Device (GR) memory	Counter	Not Applicable	600
CumulativeDBECOUNTERFB	Cumulative double-bit error counter for Frame Buffer (FB)	Counter	Not Applicable	600
CumulativeSBECOUNTERFB	Cumulative single-bit error counter for Frame Buffer (FB)	Counter	Not Applicable	600
CumulativeSBECOUNTERGR	Cumulative single-bit error counter for Graphic Device (GR)	Counter	Not Applicable	600
CumulativeDBECOUNTERGR	Cumulative double-bit error counter for Graphic Device (GR)	Counter	Not Applicable	600
DBECOUNTERFB	Double-bit errors from Frame Buffer in Frame Buffer memory	Counter	Not Applicable	600
DBECOUNTERGRL1Cache	Double-bit errors from L1 cache in Graphic Device (GR) memory	Counter	Not Applicable	600
DBECOUNTERFBL2Cache	Double-bit errors from L2 cache in Frame Buffer memory	Counter	Not Applicable	600
DBECOUNTERGRTex	Double-bit errors from texture units (TEX) in Graphic Device (GR) memory	Counter	Not Applicable	600
DBERetiredPages	Number of pages dynamically retired because of double-bit errors in Frame Buffer memory	Counter	Not Applicable	600
DBECOUNTERGRRF	Double-bit errors from SM register file (RF) in Graphic Device (GR) memory	Counter	Not Applicable	600

GPU Subsystem Power

GPU Subsystem Power metrics provide detailed insights into the power consumption of various GPU components, including standby, input, output, fan, and switch power. GPU Subsystem Power metrics provide detailed insights into the power consumption of various GPU components, including standby, input, output, fan, and switch power.

Table 29. GPU Subsystem Power

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
GPUStandbyPower	GPU standby power consumption in W	Numeric	Watts (W)	1
GPUPower	Per unit GPU power consumption in W	Numeric	Watts (W)	1
GPUInputPower	Per unit GPU input power in W	Numeric	Watts (W)	1
GPUSwitchPower	GPU Baseboard switches and redrivers power consumption in W	Numeric	Watts (W)	1
TotalGPUInputPower	Total GPU input power in W	Numeric	Watts (W)	1
GPUFanPower	Total GPU fan power consumption in W	Numeric	Watts (W)	1
TotalGPUPower	Total GPU power consumption in W	Numeric	Watts (W)	1
GPUOutputPower	GPU output power in W	Numeric	Watts (W)	1

FPGA Sensor

The FPGA Sensor offers temperature readings in Celsius with a 5-second sensing interval, offering a numeric metric type for monitoring and analysis.

Table 30. FPGA Sensor

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
Temperature Reading	Temperature sensor reading in Celsius	Numeric	Celsius (Cel)	5

PCIe Metrics

PCIe metrics describe high-speed bus standards for peripherals, offering faster speeds and flexibility, with various generations and configurations available, such as PCIe Gen 3.

PCIe Metrics offers valuable data that helps users monitor the health and performance of PCIe components in their systems. These metrics track error counts related to PCIe switch ports, including receiver errors, bad TLP (Transaction Layer Packet) errors, and bad DLLP (Data Link Layer Packet) errors. Monitoring these metrics enables users to identify potential issues with PCIe communication, ensuring smoother data transfer and improved system stability. The data can also help in diagnosing recovery errors and offer insights for preventing performance bottlenecks in high-performance environments.

Table 31. PCIe Metrics

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
CPUPower	Per socket CPU power consumption in W.	Numeric	Watts (W)	5
FPGAPower	Per FPGA Sensor Power Consumption in W.	Numeric	Watts (W)	5
TotalCPUPower	Overall CPU power consumption in W.	Numeric	Watts (W)	5
TotalFPGAPower	Overall FPGA power consumption in W.	Numeric	Watts (W)	5
TotalPciePower	Overall PCIe power consumption in W.	Numeric	Watts (W)	5
TotalMemoryPower	Overall memory power consumption in W.	Numeric	Watts (W)	5
TotalFanPower	Overall Fan power consumption in W.	Numeric	Watts (W)	5
TotalStoragePower	Overall Storage power consumption in W.	Numeric	Watts (W)	5
SystemInputPower	Current System Input Power in W.	Numeric	Watts (W)	5
SystemOutputPower	Current System Output Power in W.	Numeric	Watts (W)	5
SystemHeadRoomInstantaneous	Instantaneous Headroom in W.	Numeric	Watts (W)	5
SystemPowerConsumption	Present reading in watts in W.	Numeric	Watts (W)	5

Storage Devices

This section provides an overview of the metrics related to storage devices. This includes sensor readings and SMART (Self-Monitoring, Analysis, and Reporting Technology) data for SSD drives, SAS/SATA and NVMe (Non-Volatile Memory Express) devices. By monitoring these metrics, you can evaluate key factors such as temperature, error rates, and overall drive health. This allows for early detection of potential issues like disk failures or performance degradation. The ability to track parameters such as the remaining drive life, error counts, and read or write operations helps users proactively manage their storage devices. It also reduces downtime, and maintain high levels of system reliability and data integrity.

Storage Sensor

Table 32. Storage Sensor

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
TemperatureReading	Temperature sensor reading in Celsius	Numeric	Celsius (Cel)	5

Storage Disk SMART Data

The Storage Disk SMART Data feature shows whether SMART (Self-Monitoring Analysis and Reporting Technology) is enabled or disabled for disks, with options including Detect-Only, Enabled, and Disabled.

Table 33. Storage Disk SMART Data

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
UsedReservedBlockCount	SMART_ID_179: Count of used reserved blocks. This value is not normalized.	Counter	Not Applicable	3600
UncorrectableLBACount	SMART_ID_198: The total count of uncorrectable errors when reading/writing a sector (LBA). This value is not normalized.	Counter	Not Applicable	3600
VolatileMemoryBackupSourceFailures	SMART_ID_201: Volatile memory backup source failures. This value is not normalized	Counter	Not Applicable	3600
EraseFailCount	SMART_ID_182: Erase fail count. This value is not normalized.	Counter	Not Applicable	3600
CommandTimeout	SMART_ID_188: The number of aborted operations due to disk timeout. This value is not normalized	Counter	Not Applicable	3600
ExceptionModeStatus	SMART_ID_202: Exception mode status. This value is not normalized.	Numeric	Not Applicable	3600
UnusedReservedBlockCount	SMART_ID_180: Count of unused reserved blocks. This value is not normalized.	Counter	Not Applicable	3600
UncorrectableErrorCount	SMART_ID_187: The count of errors that could not be recovered using hardware ECC. This value is not normalized	Counter	Not Applicable	3600
DriveTemperature	SMART_ID_194: Drive temperature in Celsius. This value is not normalized.	Numeric	Celsius (Cel)	3600
ReallocatedBlockCount	SMART_ID_05: Count of reallocated blocks/sectors. The raw value represents a count of the bad blocks/sectors that have been found and remapped. This value is not normalized.	Counter	Not Applicable	3600

Table 33. Storage Disk SMART Data (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
ECCERate	SMART_ID_13: Uncorrected read errors reported to the OS. This value is not normalized.	Numeric	Not Applicable	3600
PercentDriveLifeRemaining	SMART_ID_245: Drive life remaining in percent. This value is not normalized.	Countdown	Percentage (%)	3600
PowerCycleCount	SMART_ID_12: This attribute indicates the count of full hard disk power on/off cycles. This value is not normalized	Counter	Not Applicable	3600
PowerOnHours	SMART_ID_09: The raw value of this attribute shows total count of hours (or minutes, or seconds, depending on manufacturer) in power-on state. This value is not normalized	Counter	Not Applicable	3600
MediaWriteCount	SMART_ID_233: Media (SSD) write count (a wearout indicator). This value is not normalized.	Counter	Not Applicable	3600
CurrentPendingSectorCount	SMART_ID_197: Current pending sector count. This value is not normalized	Counter	Not Applicable	3600
ReadErrorRate	SMART_ID_01: Read error rate (vendor specific). This value is not normalized.	Numeric	Not Applicable	3600
CRCErrorsCount	SMART_ID_199: The count of errors in data transfer via the interface cable as determined by ICRC (Interface Cyclic Redundancy Check). This value is not normalized	Counter	Not Applicable	3600
ProgramFailCount	SMART_ID_181: The number of times when write (program the flash) to a flash memory failed. In case of HDD, this is the number of user data accesses (both reads and writes) where LBAs are not 4 KiB	Counter	Not Applicable	3600

Table 33. Storage Disk SMART Data (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
	aligned. This value is not normalized.			

NVMe SMART Data

Table 34. NVMe SMART Data

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
DataUnitsReadLower	Specifies the lower part of the count of 512-byte data units the host has read from the controller. This value is reported in thousands and is rounded up	Counter	Blocks	3600
DataUnitsReadUpper	Specifies the upper part of the count of 512-byte data units the host has read from the controller. This value is reported in thousands and is rounded up	Counter	Blocks	3600
PercentageUsed	Specifies a vendor-specific estimate of the percentage of NVM(Non Volatile Memory) subsystem life used based on the actual usage and the manufacturer prediction of NVM life	Numeric	Percentage (%)	3600
CompositeTemperature	Indicatest the current composite temperature (in Kelvin) of the controller and namespace(s) associated with that controller	Numeric	Kelvin (K)	3600
CriticalWarning	Indicates critical warnings about the controller state	Numeric	Not Applicable	3600
AvailableSpareThreshold	The available spare value below which an asynchronous event completion may occur. The value is indicated as a normalized percentage (0 to 100%)	Numeric	Not Applicable	3600
AvailableSpare	Specifies the remaining spare capacity available as a normalized percentage (0 to 100%)	Numeric	Not Applicable	3600

Table 34. NVMe SMART Data (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
MediaDataIntegrityErrorsLower	Contains the lower part of the count of detected unrecovered data integrity errors. Includes errors such as uncorrectable ECC, CRC checksum failure, or LBA tag mismatch	Counter	Not Applicable	3600
MediaDataIntegrityErrorsUpper	Contains the upper part of the count of detected unrecovered data integrity errors. Includes errors such as uncorrectable ECC, CRC checksum failure, or LBA tag mismatch	Counter	Not Applicable	3600
DataUnitsWrittenLower	Specifies the lower part of the count of 512 byte data units the host has written to the controller. This value is reported in thousands and is rounded up	Counter	Blocks	3600
DataUnitsWrittenUpper	Specifies the upper part of the count of 512 byte data units the host has written to the controller. This value is reported in thousands and is rounded up	Counter	Blocks	3600
NumOfErrorInfoLogEntriesLower	Contains the lower part of the count of error information log entries over the life of the controller	Counter	Not Applicable	3600
NumOfErrorInfoLogEntriesUpper	Contains the upper part of the count of error information log entries over the life of the controller	Counter	Not Applicable	3600
HostReadCommandsLower	Specifies the lower part of the number of read commands completed by the controller	Counter	Not Applicable	3600
HostReadCommandsUpper	Specifies the lower part of the number of read commands completed by the controller	Counter	Not Applicable	3600
UnsafeShutdownsLower	Contains the upper part of the number of unsafe shutdowns. This count is incremented when a shutdown notification (CC.SHN)	Counter	Not Applicable	3600

Table 34. NVMe SMART Data (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
	is not received prior to loss of power			
UnsafeShutdownsUpper	Contains the lower part of the number of unsafe shutdowns. This count is incremented when a shutdown notification (CC.SHN) is not received prior to loss of power	Counter	Not Applicable	3600
HostWriteCommandsLower	Specifies the lower part of the number of write commands completed by the controller	Counter	Not Applicable	3600
HostWriteCommandsUpper	Specifies the upper part of the number of write commands completed by the controller	Counter	Not Applicable	3600
PowerOnHoursLower	Specifies the lower part of the number of hours powered on. Hours may not include time that the controller was powered and in a non-operational power state	Counter	hours (h)	3600
PowerOnHoursUpper	Specifies the upper part of the number of hours powered on. Hours may not include time that the controller was powered and in a non-operational power state	Counter	hours (h)	3600
PowerCyclesLower	Specifies the lower part of the number of power cycles	Counter	Not Applicable	3600
PowerCyclesUpper	Specifies the upper part of the number of power cycles	Counter	Not Applicable	3600
ControllerBusyTimeLower	Contains the lower part of the amount of time in minutes the controller is busy with I/O command	Counter	minutes (min)	3600
ControllerBusyTimeUpper	Contains the upper part of the amount of time in minutes the controller is busy with I/O commands	Counter	minutes (min)	3600

System and Platform-Level Monitoring

Table 35. CPU Sensors

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
TemperatureReading	CPU Temperature sensor reading in Celsius	Numeric	Celsius	5

Serial Log

Serial logs capture real-time system events and diagnostic messages transmitted through the serial interface of a device. While no specific telemetry metrics are collected at regular intervals, it will periodically stream the messages from the serial interface. These logs are critical for debugging and tracing hardware and firmware behavior during system operations or failures.

Metric Properties

These are Redfish properties (URIs) that monitors a Metric Report by specifying Metric Properties property in the Metric Definitions.

For example:

```
{
  "MetricId": null,
  "MetricProperties": [
    "/Redfish/v1/Systems/System.Embedded.1/Oem/Dell/DellNumericSensors/
iDRAC.Embedded.1_0x23_CPU1Temp#CurrentReading"
  ],
  "CollectionFunction": null,
  "CollectionDuration": null,
  "CollectionTimeScope": "Point",
  "Oem": {
    "Dell": {
      "@odata.type": "#DellMetric.v1_1_0.DellMetric",
      "CustomLabel": null,
      "FQDD": null,
      "Source": null
    }
  }
}
```

Triggers

Telemetry Triggers define conditions that impact Telemetry report streaming behavior, generating and streaming Metric Reports based on system events or user-defined conditions.

Telemetry triggers define a set of conditions that impact Telemetry report streaming behavior. Based on these conditions, the related Metric Reports are generated and streamed. The conditions can include a system event or a user-defined condition, such as a metric value crossing a threshold limit or reaching equal to a discrete value. Triggers can be configured to monitor a wide range of conditions, such as hardware failures, changes in system performance, or other significant events. iDRAC sends the associated Metric Report by using the configured streaming methods - Server-Sent Events (SSE) or Post to Subscription.

Predefined triggers are standard triggers that come with default configurations. These are generally set up for common conditions such as CPU temperature thresholds, memory usage, or power supply failures. Custom Triggers are user-defined conditions that allow for more tailored monitoring. Users can define specific metrics, conditions, and thresholds based on their individual use case, which offers more flexibility in monitoring unique system behaviors. By leveraging predefined triggers for common conditions and custom triggers for more specific or advanced scenarios, users can effectively monitor their systems and ensure timely responses to critical event.

Topics:

- [Configuring Telemetry Report Triggers Through Redfish](#)
- [Configuring Triggers Through GUI](#)
- [Custom Triggers](#)
- [Properties of Triggers](#)
- [List of Pre-Defined Triggers](#)
- [Sample Triggers Examples](#)

Configuring Telemetry Report Triggers Through Redfish

Configuring Telemetry Report Triggers Through Redfish allows customization of report generation based on specific conditions and triggers.

If you configure triggers, then it generates a new report before the scheduled report interval when trigger condition is met. By default, the system includes the triggers that are relevant for each report. You can change the trigger associations to customize when and how it generates reports based on specific conditions.

To View All the List of Triggers

To view the list of Triggers, use the command GET with the URI `/redfish/v1/TelemetryService/Triggers` and header `content-type application/json`. Telemetry triggers generates and stream reports based on error or warning conditions. You can configure them using the Redfish API or RACADM commands.

To view all the list of Triggers:

```
Command: GET
URI: /redfish/v1/TelemetryService/Triggers
Header: content-type application/json
```

For Example:

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

To View Trigger Link To Report

To view trigger link to report:

```
Command: GET
URI: /redfish/v1/TelemetryService/Triggers/<trig>
Header: content-type application/json
<trig> = CPUCriticalTrigger
```

For Example:

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

Setting a Report Trigger

Setting a report trigger configuring predefined triggers for report generation based on error or warning conditions, with options to change trigger associations and specify report intervals.

Updating a Trigger

HTTP Patch for iDRAC9 and iDRAC10:

:

```
Command: PATCH
URI: /redfish/v1/TelemetryService/MetricReportDefinitions/<MRD>
Body: {"Links": {"Triggers": [{"@odata.id": "/redfish/v1/TelemetryService/Triggers/<trig1>"}, {"@odata.id": "/redfish/v1/TelemetryService/Triggers/<trig2>"}]}}
Header: content-type application/json
<MRD> = SystemUsage
```

For Example (iDRAC9 only):

```
curl -s -k -u <user>:<password> -X PATCH https://<IDRAC_IP>/redfish/v1/TelemetryService/MetricReportDefinitions/SystemUsage -H 'Content-Type: application/json' -d '{"Links": {"Triggers": [{"@odata.id": "/redfish/v1/TelemetryService/Triggers/TMPCpuCriticalTrigger"}, {"@odata.id": "/redfish/v1/TelemetryService/Triggers/TMPCpuWarnTrigger"}]}}'
OR
n'
```

```
Command: PATCH
URI: /redfish/v1/Managers/iDRAC.Embedded.1/Attributes
Body: {": {"Telemetry<report>.1.ReportTriggers": " <trig1>, <trig2>"}
Header: content-type application/json
e.g <trig1> = CPUCriticalTrigger
    <trig2> = CPUWarntrigger
```

For Example:

```
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": {"TelemetryCPUSensor.1.ReportTriggers": "CPUCriticalTrigger, CPUWarnTrigger"}}'
```

NOTE:

The attribute feature is supported only in iDRAC9.

Deleting a Trigger

To delete a Trigger:

```
Command: Delete
Command: DELETE
URI: /redfish/v1/TelemetryService/Triggers/<Trigger_ID>
Header: content-type application/json
<Trigger_ID> = CPUWarnTrigger
```

For Example:

```
curl -s -k -u <user>:<password> -X DELETE https://<iDRAC_IP>/redfish/v1/TelemetryService/Triggers/ CPUWarnTrigger -H 'Content-Type: application/json'
```

Configuring Triggers Through GUI

Configuring triggers through iDRAC UI involves creating custom triggers and adapters to handle interactive actions and additional events.

Import Triggers

The import triggers collects data when a specific event happens and reports it as a Metric Event.

Steps

1. Go to **Configuration > System Settings > Telemetry Configuration > Triggers**.
2. Select the **Location** Type.
3. Click **Choose File** and select the file.
4. Click **Import**. This imports the trigger file.

Results

The trigger is displayed in the Triggers list.

Export Triggers

Export triggers allow for the export of telemetry triggers, which can be used to compare triggers between servers or as a template for other servers.

Steps

1. Go to **Configuration > System Settings > Telemetry Configuration > Triggers**.
2. Select the **Location** Type.
3. Click Choose File and select the file.
4. Click **Export**. This exports the Trigger file.

Results

The triggers file is displayed in the Triggers list.

Custom Triggers

Custom triggers in telemetry define conditions for generating and streaming metric reports, allowing users to monitor system events, performance changes, or significant events, and receive notifications when triggers occur.

After you configure telemetry streaming on the iDRAC, Redfishclients can either stream telemetry reports continuously or pull telemetry reports on demand. The following sections describe the methods through which clients can receive the data.

Creating a New Trigger

Define telemetry triggers by setting conditions that stream metric reports and perform actions when met, enabling automated monitoring and response.

To create a new Trigger:

```
Command: POST
URI: /redfish/v1/TelemetryService/Triggers
Body: {
  "@odata.type": "#Triggers.v1_2_0.Triggers",
  "@odata.context": "/redfish/v1/$metadata#Triggers.Triggers",
  "@odata.id": "/redfish/v1/TelemetryService/Triggers/CPUCriticalTrigger",
  "Id": "CPUTempHighWarningTrigger",
  "Name": "Trigger on CPU critical errors",
  "Description": "Trigger when an OEM event is raised",
  "TriggerActions": [
    "RedfishMetricReport"
  ],
  "TriggerActions@Redfish.AllowableValues": [
    "RedfishMetricReport"
  ],
  "EventTriggers": [
    "iDRAC.1.6.CPU0004",
    "iDRAC.1.6.CPU0700",
    "iDRAC.1.6.CPU0702",
    "iDRAC.1.6.CPU0006",
    "iDRAC.1.6.CPU0703",
    "iDRAC.1.6.CPU0701",
    "iDRAC.1.6.CPU0003"
  ],
  "Links": {
    "MetricReportDefinitions": [
      {
        "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/
CPUSensor"
      }
    ]
  }
}
Header: content-type application/json
```

For Example:

```
curl -s -k -u <user>:<password> -X POST https://<iDRAC_IP>/redfish/v1/TelemetryService/
Triggers -H 'Content-Type: application/json' -d '{
  "@odata.type": "#Triggers.v1_2_0.Triggers",
  "@odata.context": "/redfish/v1/$metadata#Triggers.Triggers",
  "@odata.id": "/redfish/v1/TelemetryService/Triggers/CPUCriticalTrigger",
  "Id": "CPUTempHighWarningTrigger",
  "Name": "Trigger on CPU critical errors",
  "Description": "Trigger when an OEM event is raised",
  "TriggerActions": [
    "RedfishMetricReport"
  ],
  "TriggerActions@Redfish.AllowableValues": [
    "RedfishMetricReport"
  ],
  "EventTriggers": [
    "iDRAC.1.6.CPU0004",
```

```

        "iDRAC.1.6.CPU0700",
        "iDRAC.1.6.CPU0702",
        "iDRAC.1.6.CPU0006",
        "iDRAC.1.6.CPU0703",
        "iDRAC.1.6.CPU0701",
        "iDRAC.1.6.CPU0003"
    ],
    "Links": {
        "MetricReportDefinitions": [
            {
                "@odata.id": "/Redfish/v1/TelemetryService/MetricReportDefinitions/
CPUSensor"
            }
        ]
    }
}
}'

```

Replacing a Trigger

If the trigger definition already exists, replace it with the new definition.

```

Command: PUT
URI: /redfish/v1/TelemetryService/Triggers/<trig_id>
Body: {
    "@odata.type": "#Triggers.v1_2_0.Triggers",
    "@odata.context": "/redfish/v1/$metadata#Triggers.Triggers",
    "@odata.id": "/redfish/v1/TelemetryService/Triggers/CPUCriticalTrigger",
    "Id": "CPUTempHighWarning2Trigger",
    "@odata.type": "#Triggers.v1_2_0.Triggers",
    "@odata.context": "/redfish/v1/$metadata#Triggers.Triggers",
    "@odata.id": "/redfish/v1/TelemetryService/Triggers/CPUCriticalTrigger",
    "Id": "CPUTempHighWarningTrigger",
    "Name": "Trigger on CPU temperature critical errors",
    "Description": "Trigger when an OEM event is raised",
    "TriggerActions": [
        "RedfishMetricReport"
    ],
    "TriggerActions@Redfish.AllowableValues": [
        "RedfishMetricReport"
    ],
    "EventTriggers": [
        "iDRAC.1.6.CPU0004",
        "iDRAC.1.6.CPU0700",
        "iDRAC.1.6.CPU0702"
    ],
    "Links": {
        "MetricReportDefinitions": [
            {
                "@odata.id": "/Redfish/v1/TelemetryService/MetricReportDefinitions/
PowerMetrics"
            }
        ]
    }
}
Header: content-type application/json
<trig_id> = CPUTempHighWarning2Trigger

```

For Example:

```

curl -s -k -u <user>:<password> -X PUT https://<IDRAC_IP>/redfish/v1/TelemetryService/
Triggers/CPUTempHighWarningTrigger -H 'Content-Type: application/json' -d ' {
    "@odata.type": "#Triggers.v1_2_0.Triggers",
    "@odata.context": "/redfish/v1/$metadata#Triggers.Triggers",
    "@odata.id": "/redfish/v1/TelemetryService/Triggers/CPUCriticalTrigger",
    "Id": "CPUTempHighWarning2Trigger",
    "Name": "Trigger on CPU temperature critical errors",
    "Description": "Trigger when an OEM event is raised",
    "TriggerActions": [
        "RedfishMetricReport"
    ],
}

```

```

    "TriggerActions@Redfish.AllowableValues": [
    "TriggerActions@Redfish.AllowableValues": [
        "RedfishMetricReport"
    ],
    "EventTriggers": [
        "iDRAC.1.6.CPU0004",
        "iDRAC.1.6.CPU0700",
        "iDRAC.1.6.CPU0702"
    ],
    "Links": {
        "MetricReportDefinitions": [
        PowerMetrics"
            { "@odata.id": "/Redfish/v1/TelemetryService/MetricReportDefinitions/"
            }
        ]
    }
}
}

```

A successful PUT operation shall provide a response with Status Code -200 OK. In case of failures it supports appropriate EEMl messages with Status Codes 400, 405.

Updating a Trigger

To update a Trigger:

```

Command: PATCH
URI: /redfish/v1/TelemetryService/Triggers/<trig_id>
BODY: {
    "TriggerActions": [
        "RedfishMetricReport"
    ],
    "TriggerActions@Redfish.AllowableValues": [
    "TriggerActions@Redfish.AllowableValues": [
        "RedfishMetricReport"
    ],
    "EventTriggers": [
        "iDRAC.1.6.CPU0004",
        "iDRAC.1.6.CPU0700",
        "iDRAC.1.6.CPU0702"
    ],
    "Links": {
        "MetricReportDefinitions": [
            {
                "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/"
            }
        ]
    }
}
Header: content-type application/json

```

For Example:

```

curl -s -k -u <user>:<password> -X PATCH https://<IDRAC_IP>/redfish/v1/TelemetryService/
Triggers/CPUTempHighWarningTrigger -H 'Content-Type: application/json' -d ' {
    "TriggerActions": [
        "RedfishMetricReport"
    ],
    "TriggerActions@Redfish.AllowableValues": [
    "TriggerActions@Redfish.AllowableValues": [
        "RedfishMetricReport"
    ],
    "EventTriggers": [
        "iDRAC.1.6.CPU0004",
        "iDRAC.1.6.CPU0700",
        "iDRAC.1.6.CPU0702"
    ],
    "Links": {

```

```

        "MetricReportDefinitions": [
            {
                "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/
PowerMetrics"
            }
        ]
    }
}
'
        "MetricReportDefinitions": [
            {
                "@odata.id": "/redfish/v1/TelemetryService/MetricReportDefinitions/
PowerMetrics"
            }
        ]
    }
}
'

```

A successful PATCH operation shall provide a response with Status Code -200 OK. In case of failures it supports appropriate EEMI messages with Status Codes 400, 405.

Deleting a Trigger

To delete a Trigger:

```

Command: Delete
URI: /redfish/v1/TelemetryService/Triggers/<Trigger_ID>
Header: content-type application/json
<Trigger_ID> = CPUTempHighWarningTrigger

```

For Example:

```

curl -s -k -u <user>:<password> -X DELETE https://<IDRAC_IP>/redfish/v1/TelemetryService/
Triggers/CPUTempHighWarningTrigger -H 'Content-Type: application/json'

```

Properties of Triggers

Triggers have properties like DiscreteTriggerCondition, DiscreteTriggers, EventTriggers, MetricProperties, MetricType, and TriggerActions. These properties define conditions and actions for triggers.

The following are the properties of Triggers:

- **MetricType** – This determines the type of the trigger (discrete, or numeric) and corresponding specifications (EventTriggers, DiscreteTriggers, or NumericThresholds). The specifications and type must match and not be mixed.
 - Omitted (not present) - Fired only on events listed in EventTriggers
 - Numeric – The trigger is for numeric sensor. Uses Numeric Thresholds to fire the trigger
 - Discrete – trigger is fired when a metric value is exactly equal to a specified value or change of value detected.
- **DiscreteTriggerCondition**
 - Specified - A discrete trigger condition is met when the metric value becomes one of the values that the DiscreteTriggers property lists.
 - Changed - A discrete trigger condition is met whenever the metric value changes.
- **DiscreteTriggers** - This property shall contain a list of values to which to compare a metric reading. This property shall be present when the DiscreteTriggerCondition property is Specified.
 - DwellTime - This property shall contain the amount of time that a trigger event persists before the MetricAction is performed. The fractional seconds are ignored.
 - Value - This property shall contain the value discrete metric that constitutes a trigger event. The DwellTime shall be measured from this point in time.
 - Severity – ignored
 - Name – ignored
- **EventTriggers** – Array of message IDs to trigger on. If any event is raised with this message id, this trigger is fired
- **Links/MetricReportDefinitions** – List of all the Reports that are generated when this trigger fires.
- **MetricProperties** – See [Metric Report Definitions \(MRD\)](#) and [Metric Report](#)
- **Wildcards** – See [Metric Report Definitions \(MRD\)](#)

- MetricIds – This shall contain an array of MetricIds whose numeric or discrete value is used to evaluate the threshold crossing or discrete trigger condition. It is a many-to-many relation between MetricIds and numeric threshold “Reading”s or discrete trigger “Value”s. Note: Only either MetricIds or MetricProperties with optional Wildcards is supported, not both.
- NumericThresholds
 - LowerCritical/LowerWarning/UpperCritical/UpperWarning
 - Activation
 - Decreasing – Trigger when the metric value starts higher than, then falls below the specified Reading and remains below for the specified DwellTime
 - Either – Fire the trigger any time the value crosses from above to below or below to above the Reading for DwellTime duration.
 - Increasing – fire the trigger when the metric value starts lower than, then rises above the specified Reading and remains above for the specified DwellTime.
 - DwellTime - This property shall indicate the duration the sensor value must violate the threshold before the threshold is activated. The fractional seconds are ignored.
 - Reading - The numeric value for comparison
- TriggerActions
 - RedfishMetricReport - This value indicates that when a trigger condition is met, the service shall force the metric reports managed by the MetricReportDefinitions specified by the MetricReportDefinitions property to be updated, regardless of the MetricReportDefinitionType property value. The actions specified in the ReportActions property of each MetricReportDefinition shall be performed.
- Name – The name of the trigger.

List of Pre-Defined Triggers

The provided text lists various pre-defined triggers, including CPUCriticalTrigger, CPUWarnTrigger, and others, used for report generation in iDRAC Telemetry. These triggers are related to different components such as CPU, memory, and thermal sensors, used to detect critical or warning events.

The following are the list of Pre-defined triggers :

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

Sample Triggers Examples

The provided text describes two types of triggers in Redfish telemetry service: Discrete Trigger and Numeric Trigger, each with its own set of properties and conditions, such as metric type, trigger actions, and threshold values.

Discrete Trigger

```
{
  "@odata.type": "#Triggers.v1_1_1.Triggers",
  "@odata.id": "/Redfish/v1/TelemetryService/Triggers/CustomDiscreteTrigger",
```

```

    "Id": "CustomDiscreteTrigger",
    "Name": "Custom Discrete Trigger",
    "Description": "Custom Discrete Trigger",
    "MetricType": "Discrete",
    "TriggerActions": [
        "RedfishMetricReport "
    ],
    "DiscreteTriggerCondition": "Specified",
    "DiscreteTriggers": [
        {
            "Value": "Non-operational",
            "DwellTime": "PT0.001S",
            "Severity": "Warning"
        }
    ],
    "MetricIds": [
        "OSDriverState"
    ],
    "Links": {
        "MetricReportDefinitions":
[
        {
            "@odata.id": "/Redfish/v1/TelemetryService/MetricReportDefinitions/
NICStatistics"
        } ]}}

```

Numeric Trigger

```

{
    "@odata.type": "#Triggers.v1_1_1.Triggers",
    "@odata.id": "/Redfish/v1/TelemetryService/Triggers/CustomNumericTrigger",
    "Id": "CustomNumericTrigger",
    "Name": "Custom Numeric Trigger",
    "Description": "Custom Numeric Trigger",
    "MetricType": "Numeric",
    "TriggerActions": [
        "RedfishMetricReport"
    ],
    "NumericThresholds": {
        "UpperCritical": {
            "Reading": 70,
            "Activation": "Increasing",
            "DwellTime": "PT0.001S"
        },
        "UpperWarning": {
            "Reading": 65,
            "Activation": "Increasing",
            "DwellTime": "PT0.004S"
        }
    },
    "MetricIds": [
        "TotalCPUPower"
    ],
    "Links": {
        "MetricReportDefinitions": [
            {
                "@odata.id": "/Redfish/v1/TelemetryService/MetricReportDefinitions/
PowerMetrics"
            },
            {
                "@odata.id": "/Redfish/v1/TelemetryService/MetricReportDefinitions/
CPUSensor"
            }
        ]
    }
}

```

Metric Injection

The Metric Injection feature in iDRAC Service Module (iSM) collects metrics from the operating system and injects the metrics data into the iDRAC telemetry interface, supporting static and dynamic metrics with a default update interval of three minutes.

There is one configurable characteristic for metric injection feature is to enable or disable external event injection. By default, it is disabled.

EnableMetricInjection - If the feature is enabled, external agents (iSM or other) can use the interface provided to add new Metric Definitions (that is, MetricIDs) and then to repeatedly report values for these new metrics over time, and the user can add these new MetricIDs to a custom MetricReportDefinition to get metric reports with the new metrics. Currently metric injection is allowed inband using USB NIC interface only.

Topics:

- [Enabling Metric Injection Feature](#)
- [Disabling Metric Injection Feature](#)
- [View current status of metric injection](#)
- [OS Metrics](#)

Enabling Metric Injection Feature

The Metric Injection (Host OS Telemetry) feature in iDRAC Service Module (iSM) collects metrics from the operating system and injects the metrics data into the iDRAC telemetry interface.

Enabling metric injection feature using RACADM

Run the following command to enable the metric injection feature using RACADM:

```
racadm set iDRAC.Telemetry.EnableMetricInjection Enable
```

Enable metric injection using Redfish

Enable metric injection using Redfish with a PATCH command and JSON body.

To enable metric injection using Redfish:

```
Command: PATCH
URI: /redfish/v1/Managers/iDRAC.Embedded.1/Attributes
Body: {"Attributes": {"Telemetry.1.EnableMetricInjection": "Enabled"}}
Header: content-type application/json
```

For Example:

```
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": {"Telemetry.1.EnableMetricInjection": "Enable"}}'
```

Disabling Metric Injection Feature

Disable metric injection using RACADM

Run the following command to disable Metric Injection using RACADM:

```
racadm set iDRAC.Telemetry.EnableMetricInjection Disable
```

Disable metric injection using Redfish

To disable metric injection using Redfish:

```
Command: PATCH
URI:/redfish/v1/Managers/iDRAC.Embedded.1/Attributes
Body: {"Attributes": {"Telemetry.1.EnableMetricInjection": "Disabled"}}
Header: content-type application/json
```

For Example:

```
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": {"Telemetry.1.EnableMetricInjection": "Disable"}}'
```

View current status of metric injection

To view the status of metric injection, run the following command:

```
racadm get iDRAC.Telemetry.EnableMetricInjection
```

NOTE:

- Only users with iDRAC Admin privileges can enable metric injection.
- The metric injection feature is only available to OS application via inband using OAuth token.
- New MetricDefinition (MD) can only be added when MetricInjection is enabled.
- New MD can be added or deleted and cannot be modified.
- Currently a maximum of 50 MDs can be injected.
- The injected MDs will be persisted available after iDRAC reset, similar to MRDs.
- `racresetcfg` `racadm` command removes the injected MDs.
- Submit metric values is only allowed when Metric Injection is enabled.
- Submit metric values will be allowed without nonexisting metric id but the report will not have those instances.
- The report generation and streaming behavior for the injected metrics will be the same as existing reports today.

OS Metrics

OS metrics are injected by the Dell iDRAC Service Module (iSM) software, if installed. iSM releases may add additional metric support in the upcoming releases. For the updated metric list and details, see the latest iSM user guide.

The Metric Injection (Host OS Telemetry) feature collects static and dynamic metrics from the host operating system. The static metrics are injected only when the iSM service starts or restarts. The dynamic metrics have a default update interval of three minutes.

Table 36. OS Metrics

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
OSProcessorMaxFrequency	The maximum value for the Processor Frequency	Static Numeric	Giga-Hertz (GHz)	3600
OSNumberOfProcessorCores	The total number of cores	Static Numeric	Not Applicable	3600
OSProcessorArchitecture	The Processor Architecture type	Static Discrete	Not Applicable	3600
OSTotalPhysicalMemory	The total size of visible physical memory	Static Numeric	Giga-Bytes (GB)	3600
OSTotalVirtualMemory	The total size of virtual memory.	Static Numeric	Giga-Bytes (GB)	3600
OSProcessorUtilizationPercentage	The current Percentage of Processor Utilization	Dynamic Numeric	Percentage (%)	180(Default)
OSProcessorOperatingFrequency	The current value for the Operating Frequency of the Processor	Dynamic Numeric	Giga-Hertz (GHz)	180(Default)
OSNumberOfProcesses	The current Number of Processes running on the operating system	Dynamic Numeric	Not Applicable	180(Default)
OSFreePhysicalMemory	The size of the free physical memory.	Dynamic Numeric	Giga-Bytes (GB)	180(Default)
OSFreeVirtualMemory	The size of the free virtual memory.	Dynamic Numeric	Giga-Bytes (GB)	180(Default)
OSMemoryUtilizationPercentage	The percentage of the used physical memory	Dynamic Numeric	Percentage (%)	180(Default)
OSPhysicalDriveNodeName	The name of the node to which this physical drive is attached.	Dynamic Discrete	Not Applicable	1800(Default)
OSPhysicalDriveDiskName	The name of the physical drive.	Dynamic Discrete	Not Applicable	1800(Default)
OSPhysicalDriveDiskSerial	The disk serial of the physical drive	Dynamic Discrete	Not Applicable	1800(Default)
OSPhysicalDriveBusType	The bus type for the physical drive	Dynamic	Not Applicable	1800(Default)

Table 36. OS Metrics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
		Discrete		
OSPhysicalDriveDiskMediaType	The media type for the physical drive.	Dynamic Discrete	Not Applicable	1800(Default)
OSPhysicalDriveDiskId	The disk Id for the physical drive.	Dynamic Numeric	Giga-Bytes (GB)	1800(Default)
OSPhysicalDriveDiskSize	The disk size for the physical drive	Dynamic Numeric	Giga-Bytes (GB)	1800(Default)
OSPhysicalDriveDiskAllocated	The disk allocated size for the physical drive	Dynamic Discrete	Giga-Bytes (GB)	1800(Default)
OSPhysicalDriveCanPool	Indicates if the Physical Drive is available to be part of a Storage Pool.	Dynamic Discrete	Not Applicable	1800(Default)
OSPhysicalDriveCannotPoolReason	Reason why the Physical Drive is not eligible for pooling	Dynamic Discrete	Not Applicable	1800(Default)
OSPhysicalDriveVirtualDiskFootprint	The physical drive Virtual Disk Footprint	Dynamic Numeric	Not Applicable	1800(Default)
OSLogicalDriveDriveLetter	The drive letter for the Logical Drive.	Dynamic Discrete	Not Applicable	1800(Default)
OSLogicalDriveName	The drive name for the Logical Drive	Dynamic Discrete	Not Applicable	1800(Default)
OSLogicalDriveFileSystemType	The file system type for the Logical Drive	Dynamic Discrete	Not Applicable	1800(Default)
OSLogicalDriveDriveType	The drive type for the Logical Drive.	Dynamic Discrete	Not Applicable	1800(Default)
OSLogicalDriveHealthStatus	The drive health status for the Logical Drive	Dynamic Discrete	Not Applicable	1800(Default)
OSLogicalDriveOperationalStatus	The drive operational status for the Logical Drive	Dynamic Discrete	Not Applicable	1800(Default)
OSLogicalDriveFreeSpace	The amount of free space on the Logical Drive.	Dynamic Numeric	Giga-Bytes (GB)	1800(Default)

Table 36. OS Metrics (continued)

Metric Ids	Description	Metric Type	Units	Sensing Interval in seconds
OSLogicalDriveTotalSize	The amount of total space on the Logical Drive	Dynamic Numeric	Not Applicable	1800(Default)
OSLogicalProcessorUtilizationPercentage	The Logical Processor Utilization Percentage	Dynamic Numeric	Not Applicable	900(Default)

Server Configuration Profile (SCP)

Topics:

- [Telemetry Configuration Using Server Configuration Profile \(SCP\)](#)
- [SCP Export](#)
- [SCP Import](#)

Telemetry Configuration Using Server Configuration Profile (SCP)

Server Configuration Profiles (SCP) are XML or JSON templates that contain configuration settings for an individual server. Each configurable setting is a simplified name-value pair. SCP templates offer a convenient advantage by consolidating all the settings and options in a single, easily readable, and editable template that can be reapplied to any number of setups.

SCP Export

The SCP export is the process of generating an XML or JSON template. This template can be exported locally or to a network share. SCP Export includes an optional parameter called "IncludeCustomTelemetry". The Optional parameters expand the default template behavior. When the new optional parameter is selected for SCP Export, the Telemetry Custom definitions are applied from the SCP XML/JSON template.

The following is an example of a simplified XML, exported using SCP with the "IncludeCustomTelemetry" option. The CustomComponents node includes the telemetry custom definitions.

```
<CustomComponents>
  <CustomMetricReportDefinitions>
    <odata.type>#MetricReportDefinition.v1_4_2.MetricReportDefinition</odata.type>
    <odata.context>/
Redfish/v1/$metadata#MetricReportDefinition.MetricReportDefinition</odata.context>
    <odata.id>/Redfish/v1/TelemetryService/MetricReportDefinitions/AggregationMetrics</
odata.id>
    <Id>AggregationMetrics</Id>
    <Name>Aggregation Metrics Metric Report</Name>
    <Description>Aggregation Metrics report is derived by applying a formula or
filter to a set of base metric values for Power, Temperature, CUPS (CPU, Memory, IO,
System)</Description>
    <AppendLimit>2400</AppendLimit>
    <MetricReportDefinitionEnabled>true</MetricReportDefinitionEnabled>
    <MetricReportDefinitionType>OnRequest</MetricReportDefinitionType>
    <MetricReportDefinitionType.Redfish.AllowableValues>Periodic</
MetricReportDefinitionType.Redfish.AllowableValues>
    <MetricReportDefinitionType.Redfish.AllowableValues>OnChange</
MetricReportDefinitionType.Redfish.AllowableValues>
    <MetricReportDefinitionType.Redfish.AllowableValues>OnRequest</
MetricReportDefinitionType.Redfish.AllowableValues>
    <MetricReportHeartbeatInterval>PT0H0M0S</MetricReportHeartbeatInterval>
    <SuppressRepeatedMetricValue>false</SuppressRepeatedMetricValue>
    <ReportTimespan>PT0H2M0S</ReportTimespan>
    <ReportUpdates>AppendWrapsWhenFull</ReportUpdates>
    <ReportUpdates.Redfish.AllowableValues>AppendStopsWhenFull</
ReportUpdates.Redfish.AllowableValues>
    <ReportUpdates.Redfish.AllowableValues>AppendWrapsWhenFull</
ReportUpdates.Redfish.AllowableValues>
    <ReportUpdates.Redfish.AllowableValues>NewReport</
ReportUpdates.Redfish.AllowableValues>
```

```
<ReportUpdates.Redfish.AllowableValues>Overwrite</
ReportUpdates.Redfish.AllowableValues>
```

The following is an example of a simplified JSON, exported using SCP where custom Telemetry definitions are included. The "CustomComponents" node includes the Telemetry custom definitions.

```
"CustomComponents": {
  "CustomMetricReportDefinitions": [
    {
      "@odata.type":
"#MetricReportDefinition.v1_4_2.MetricReportDefinition",
      "@odata.context": "/",
Redfish/v1/$metadata#MetricReportDefinition.MetricReportDefinition",
      "@odata.id": "/Redfish/v1/TelemetryService/MetricReportDefinitions/
AggregationMetrics",
      "Id": "AggregationMetrics",
      "Name": "Aggregation Metrics Metric Report",
      "Description": "Aggregation Metrics report is derived by applying
a formula or filter to a set of base metric values for Power, Temperature, CUPS (CPU,
Memory, IO, System)",
      "AppendLimit": 2400,
      "MetricReportDefinitionEnabled": true,
      "MetricReportDefinitionType": "OnRequest",
      "MetricReportHeartbeatInterval": "PT0H0M0S",
      "SuppressRepeatedMetricValue": false,
      "ReportTimespan": "PT0H2M0S",
      "ReportUpdates": "AppendWrapsWhenFull",
      "Wildcards": [],
      "MetricReportDefinitionType@Redfish.AllowableValues": [
        "Periodic",
        "OnChange",
        "OnRequest"
      ],
      "ReportUpdates@Redfish.AllowableValues": [
        "AppendStopsWhenFull",
        "AppendWrapsWhenFull",
        "NewReport",
        "Overwrite"
      ],
      "ReportActions": [
        "LogToMetricReportsCollection"
      ],
      "ReportActions@Redfish.AllowableValues": [
        "LogToMetricReportsCollection",
        "RedfishEvent"
      ],
      "Status": {
        "State": "Enabled"
      },
      "Schedule": {
        "RecurrenceInterval": "PT0H0M0S"
      },
      "MetricReport": {
        "@odata.id": "/Redfish/v1/TelemetryService/MetricReports/
AggregationMetrics"
      },
      "Metrics": [
        {
          "MetricId": "SystemAvgInletTempHour",
          "MetricProperties": [],
          "CollectionFunction": null,
          "CollectionDuration": null,
          "CollectionTimeScope": "Point",
          "Oem": {
            "Dell": {
              "@odata.type": "#DellMetric.v1_1_0.DellMetric",
              "CustomLabel": null,
              "FQDD": null,
              "Source": null
            }
          }
        }
      ],
    }
  ]
}
```

```

        "MetricId": "SystemMaxInletTempHour",
        "MetricProperties": [],
        "CollectionFunction": null,
        "CollectionDuration": null,
        "CollectionTimeScope": "Point",
        "Oem": {
            "Dell": {
                "@odata.type": "#DellMetric.v1_1_0.DellMetric",
                "CustomLabel": null,
                "FQDD": null,
                "Source": null
            }
        }
    },
    {
        "MetricId": "SystemMaxPowerConsumption",
        "MetricProperties": [],
        "CollectionFunction": null,
        "CollectionDuration": null,
        "CollectionTimeScope": "Point",
        "Oem": {
            "Dell": {
                "@odata.type": "#DellMetric.v1_1_0.DellMetric",
                "CustomLabel": null,
                "FQDD": null,
                "Source": null
            }
        }
    }
],
"Links": {
    "Triggers": []
},
"Oem": {
    "Dell": {
        "@odata.type":
"#DellMetricReportDefinition.v1_1_0.DellMetricReportDefinition",
        "Digest":
"45ab1831e9f814eaae29bdc44b55fabf197fb266baffc5e14df2a842875335d2",
        "iDRACFirmwareVersion": "1.20.50.50"
    }
}
}

```

SCP Export using Redfish

To export the SCP profile using Redfish, perform a POST method on:

```

Command: POST
iDRAC9 URI: /redfish/v1/Managers/iDRAC.Embedded.1/Actions/Oem/
EID_674_Manager.ExportSystemConfiguration

iDRAC10 URI:
/redfish/v1/Managers/iDRAC.Embedded.1/Actions/Oem/OemManager.ExportSystemConfiguration
Header: content-type application/json

```

JSON Format Redfish Post example:

```

Body: {s"ExportFormat":"JSON","ExportFormat":"XML","IncludeInExport":
["IncludeCustomTelemetry"],"ShareParameters": {"Target": ["ALL"]}}

```

XML Format Post example:

```

Body: {"ExportFormat":"XML","ExportFormat":"XML","IncludeInExport":
["IncludeCustomTelemetry"],"ShareParameters":{"Target": ["ALL"]}}

```

For more information on how to download the exported file, see Section 23: Configuration/Server Configuration Profile of [iDRAC-User-Interface-to-Redfish-Mapping](#).

SCP Export using Remote RACADM

Run the RACADM get command using the `-t xml` or `-t json` parameters to invoke the SCP Export operation. Use the `includeCustomTelemetry` option to include custom Telemetry configuration in the exported template.

For example:

```
C:\>racadm -r <iDRAC IP> -u <username> -p <user password> get -f <template file name> -t xml -l <NFSshare IP>:/nfs --includeCustomTelemetry
```

SCP Export using UI

About this task

To export an SCP profile using iDRAC UI:

Steps

1. Go to **Configuration > Server Configuration Profile**.
2. Expand the **Export** option.
3. Select or enter the required information.
4. Select the Custom Telemetry checkbox to include custom Telemetry configuration into the exported template and click **Export**.

SCP Import

SCP Import is a process where server configuration is applied using an XML or JSON template. Custom Telemetry Configuration is optional and is only applied if it is in the template. SCP operations create a Job that contains information about the status of the operation. The job can be identified by a Job Identifier (JID) that is returned when an SCP operation is created. All iDRAC jobs can be found in the iDRAC UI, under **Maintenance > Job Queue** or through any other iDRAC interface.

The following is the RACADM example:

```
racadm jobqueue view -i JID_952589510966
Output:
----- JOB -----
[Job ID=JID_952589510966] Job Name=Configure: Import Server Configuration Profile
Status=Completed Scheduled Start Time= [Not Applicable] Expiration Time=[Not Applicable]
Actual Start Time=[Mon, 20 Jul 2020 10:29:11] Actual Completion Time=[Mon, 20 Jul
2020 10:29:13] Message=[SYS053: Successfully imported and applied Server Configuration
Profile.]
Percent Complete= [100]
-----
```

SCP Import using Redfish

To import the SCP profile using Redfish, perform a POST method on OEM action EID_674_Manager. If the importing template includes Telemetry custom definitions, then those configuration settings are applied.

For example:

```
Command: POST

iDRAC9 URI: redfish/v1/Managers/iDRAC.Embedded.1/Actions/Oem/
EID_674_ImportSystemConfiguration

iDRAC10 URI: redfish/v1/Managers/iDRAC.Embedded.1/Actions/Oem/
OemManager.ImportSystemConfiguration

Header: content-type application/json
```

```
Auth: Basic or X auth
Body:{"ImportBuffer": "<pass in the complete SCP File as a string value",
"ShareParameters": {"Target": ["ALL"]}}
```

For more information about how to pass the body content, see Section 23: Configuration/Server Configuration Profile of [iDRAC-User-Interface-to-Redfish-Mapping](#).

SCP Import using Remote RACADM

Run the RACADM set command using the `-t xml` or `-t json` parameters to invoke an SCP Import operation. If the importing template includes Telemetry custom definitions, those configuration settings are applied.

For example:

```
RACADM example:
C:\> >racadm -r <iDRAC IP> -u <username> -p <user password> set -f <template file name>
-t xml -l <NFS server IP>:/nfs
```

SCP Import using iDRAC UI

About this task

To import SCP using iDRAC UI:

Steps

1. Go to **Configuration > Server Configuration Profile**.
2. Expand the **Import** option.
3. Select or enter the required information.
4. Click **Import**.

If the importing template includes Telemetry custom definitions, those configuration settings are applied.

Historical Temperature Sensor Reports

The Historical Temperature Sensor Reports feature allows users to access sensor data for diagnostic purposes.

iDRAC Telemetry sends time series sensors, performance, and other data to the clients who subscribed to the data, or when the data is requested. This data is in the form of reports in JSON format which has a limited size and a time period (up to 2 hours). The historical data is not maintained on iDRAC, and users are responsible for storing the large historical data for analytical and diagnostic purposes. The data is not stored on iDRAC persistently.

This feature addresses the specific needs of making the historical sensor data available for approximately 1 month for diagnostics purposes. Clients like SupportAssist collectors can get that data from iDRAC to help diagnose problems that are related to server components. This is achieved by storing the time series data, already being collected as part of Telemetry streaming, persistently on eMMC in an efficient fashion so that it takes less space and causes less wear to eMMC and is conveniently accessible by the users.

The historical temperature sensor data can be accessed using the Redfish API or through SupportAssist Collector.

NOTE:

Datacenter license is required to get the historical temperature sensor data through Redfish interface.

Topics:

- [Getting Historical Temperature Sensor Reports Through SupportAssist Collector](#)
- [Getting Historical Temperature Sensor Reports Through Redfish](#)

Getting Historical Temperature Sensor Reports Through SupportAssist Collector

Select the “Debug Logs” option in the SupportAssist collector interface to include the historical temperature sensor data.

To find historical compressed data archive:

1. Go to the tsr compressed folder in your local file system, and extract the file.
2. Go to tsr > dbglog > logs > D4D_Temperature_archive_2024-08-07 15_00_34.84 -0500 CDT_to_2025-02-06 13_46_00.35 -0600 CST.zip
3. Open the .zip folder to view all the TemperaturesSensorD4D files. The data is collected for every 2 hours during the period that is specified by the archive file.

Getting Historical Temperature Sensor Reports Through Redfish

The historical data consists of reports, each saved with a timestamp appended to its name. The collection can be retrieved through a GET request on the following URL.

```
Command: GET
URI:
To view the Report Definition
/redfish/v1/TelemetryService/Oem/Dell/D4D/ReportDefinitions
To view the single Report Definition
/redfish/v1/TelemetryService/Oem/Dell/D4D/ReportDefinitions/TemperatureSensorD4D

To view all the Reports
/redfish/v1/TelemetryService/Oem/Dell/D4D/Reports
```

To view the single Report Data

```
/redfish/v1/TelemetryService/Oem/Dell/D4D/Reports/  
TemperatureSensorD4D-2021-06-11T15:01:00-05:00" (example)
```

Multiple reports can be queried using the start and end time query parameter. The query returns a year worth of temperature sensor historical data from the iDRAC.

 **NOTE:** The date range provides granularity for up to 2 hours.

You can perform a GET method on `/api/TemperatureSensorD4D` and save the file in .zip format. Data will be available as a compressed tar archive. The HTTP response header shall denote the content type as `application/x-tar`. The file can be saved to the local system and extracted.

```
Header: content-type application/json  
Auth: Basic or X auth
```

Telemetry Reference Tools

The Telemetry Reference Tool is an open source tool that is published at [GitHub](#), that runs outside iDRAC and provide docker containers to facilitate few steps to ingest iDRAC telemetry into the timeseries databases like Elasticsearch and InfluxDB. It can also be integrated with visualization solutions like Kibana, Grafana, and Splunk Analytics.

The basic workflow of iDRAC Telemetry Reference Tools includes:

1. Telemetry data collection from one or more iDRACs.
2. Ingesting the data into big databases like Elasticsearch, Prometheus, Timescale, and Influx DB.
3. Visualizing the data using visualization tools like Kibana, Grafana, or Splunk Analytics.

The following are the requirements for using the Telemetry Reference Tools:

1. iDRAC with firmware 4.00.00.00 or later versions and with Datacenter license.
2. Windows or Linux system, like Ubuntu OS version 20.04.1 LTS or later, with at least 16 GB RAM and 250GB available disk space .
3. Docker and Docker-Compose tools 2.2.0 or higher.
4. Remote Desktop connection to the Linux system or equivalent if GUI interfaces are used.
5. Basic knowledge of Splunk and the Open-Source tools like ELK stack (Elastic Search and Kibana), InfluxDB, Prometheus, and Grafana.

For more information, see

[GitHub - dell/iDRAC-Telemetry-Reference-Tools: Reference toolset for PowerEdge Telemetry metric collection and integration with analytics and visualization solutions.](#)

[iDRAC9-Telemetry-Reference-Tools---Data-Visualization](#)

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 have different numbers of metric values in the resulting reports that match the rate at which the backend daemons report data for these metrics.

NOTE: There may be a variance in Metric Value count variance that appear in a report above what you might expect by doing the math of $\text{RecurrenceInterval} / \text{SensorInterval}$, as the rate at which metrics are ingested is clocked by backend services reporting the metrics. 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 that are configured with no suppress repeated metrics option stream last read data if the server is powered off.

Custom reports are limited to a total of 50 total report definitions, including 28 predefined report definitions and potentially 22 custom report definitions.

Metrics in an MRD is limited to Maximum of 64 Metric IDs per report and reports are limited to 2400 metric values per report.

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.
- StorageSensor report is only supported for the drives in nonraid mode and not behind the BOSS controller.
- GPUStatistics report is only available in specific GPU models that support ECC memory capability (GP102GL [Tesla P40]).
- Platforms with ME, SiEN, and SKU which do not support the Compute Usage Per Second (CUPS) feature.
- The CUPS feature is not supported on PowerEdge R930, R240, R340, T140, and T340 systems.
- Memory, I/O, and SYS Usage not available for systems with AMD processors.
- FanSensor reports are generated only for Monolithic servers. For modular servers, the report is empty (with "MetricValues@odata.count": 0).
- When the 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 is set to 0s, the report can only be pulled and it cannot be streamed. The report is single instance and nonrepeating data, if available at the 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 was 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.

Topics:

- [Feature Constraints](#)
- [iDRAC10 Telemetry Feature Changes and Limitations](#)
- [Behavior of Telemetry feature](#)

Feature Constraints

The following are the constraints:

1. iDRAC Telemetry supports only HTTPS-based communication to the client.
2. iDRAC Telemetry supports up to eight subscriptions.

3. Deletion of subscriptions is supported through Redfish interface only, even for the manual deletion by the Admin.

iDRAC10 Telemetry Feature Changes and Limitations

The iDRAC10 Telemetry has the following feature Changes and limitations:

- The remote Syslog option for telemetry streaming is not supported in iDRAC9 firmware version 7.00.00.00 and higher. It is also not supported in iDRAC10.
- The telemetry report configuration using racadm has been removed on iDRAC10.
- The predefined reports CPURegisters and SerialLog are not supported on iDRAC10

Behavior of Telemetry feature

The telemetry feature behavior is as follows:

- iDRAC generates and pushes (HTTP POST) the Metric Report to all the subscribed clients to the destination specified in the subscription. The clients receive new data only upon successful subscription creation.
- The metric data includes the timestamp in ISO format, UTC time (ends in 'Z'), at the time of data collection from the source.
- Clients can terminate a subscription by sending an HTTP DELETE message to the URI of the subscription resource through the Redfish interface.
- If the subscription is deleted either by iDRAC or the client, then iDRAC does not send (HTTP POST) reports. If the number of delivery errors exceeds predefined thresholds, then iDRAC may delete a subscription.
- If a user has Admin privilege, they can delete the subscriptions but only through Redfish interface.
- The client is notified about the termination of a subscription by iDRAC by sending the 'Subscription terminated' event as the last message.
- Subscriptions are persistent and can remain even after iDRAC restarts. However, the subscriptions will be removed by performing racresetcfg or racadm systemerase iDRAC operations.
- User interfaces like RACADM, Redfish, SCP, and iDRAC UI display the status of the client subscriptions.
- TelemetryService readiness can be checked using a new attribute `TelemetryServiceStatus` added under the `GetRemoteServiceAPIStatus` API call. This attribute is added to the existing list of `LTStatus`, `RTStatus`, `ServerStatus`, and `Status`.

Best practices

The following are the best practices:

1. It is recommended to use a Server Configuration Profile (SCP) to configure all metric reports by enabling the Custom Telemetry option. Once created, the SCP file can be applied to multiple servers that support the Telemetry feature and have a Datacenter license.
2. Configure the report interval based on the system configuration and number of configured Telemetry reports. On a max configuration system, a high report interval (2 hours) can in-turn result in large Telemetry reports, since it includes every relevant device metric. Also, a minimum report interval (10 s) can in-turn contribute to processing overheads based on the number active configured reports.
3. For servers with max configurations (a large number of hard drives or memory cards), it is recommended to not set the `RecurrenceInterval` to maximum value.
4. For reports like `SystemUsage`, `PowerMetrics`, `CPUMemMetrics`, `ThermalMetrics`, and `GPUMetrics`, it is recommended to set a minimum `RecurrenceInterval` of 60 s, even though the minimum value allowed for `ReportInterval` is 10 s.

Troubleshooting Tips

The following table shows the most commonly seen issues and their possible causes when using Telemetry:

Table 37. Issues and their possible causes

Issues	Possible Causes	Solution
POST, PATCH operations failure.	Service is not enabled. Property "ServiceEnabled" is set to false. Applies to /Redfish/v1/EventService	1. Set the Service property "ServiceEnabled" to true. 2. Check LC logs For more information, see Enabling Global Telemetry through Redfish
	Property that is added in the input payload is not allowed or the value that is added is invalid.	Check Redfish documentation for allowed properties and valid values. For more information, see Section 6.134.3 in Redfish Schema Guide
	User account without "Administrator" privilege	1. Ensure that the user account has "Administrator" privilege. 2. Check LC logs.
DELETE operations failure.	Service is not enabled. Property "ServiceEnabled" is set to false. Applies to Redfish/v1/EventService	1. Set the Service property "ServiceEnabled" to true. 2. Check LC logs For more information, see Enabling Global Telemetry through Redfish
GET Metric Report failure.	Required license is not installed, or installed datacenter license is expired	1. Install a new license. 2. Check LC logs. For more information, see PowerEdge: How to Import and Export an iDRAC License Using the Web Interface
	Service is not enabled. Property "ServiceEnabled" is set to false. Applies to Redfish/v1/EventService	1. Set the Service property "ServiceEnabled" to true. 2. Check LC logs. For more information, see Enabling Global Telemetry through Redfish
No Metric Report in the SSE or subscription stream	Service is not enabled. Property "ServiceEnabled" is set to false. Applies to Redfish/v1/EventService	1. Set the Service property ServiceEnabled to true. 2. Check LC logs. For more information, see Enabling Global Telemetry through Redfish
	Metric report definition (MRD) is not enabled	1. Set the MRD property MetricReportDefinitionEnabled to true. 2. Check LC logs. For more information, see Enabling Metric Report Definition Using Redfish
	MRD property "ReportActions" does not have "RedfishEvent"	PATCH MRD to have RedfishEvent within ReportActions.

Table 37. Issues and their possible causes (continued)

Issues	Possible Causes	Solution
		For more information, see Configurable Properties in MRD and Enabling Metric Report Definition Using Redfish
	The required license is not installed, or installed datacenter license is expired.	1. Install a new license. 2. Check LC logs. For more information, see PowerEdge: How to Import and Export an iDRAC License Using the Web Interface
	No network route to client.	1. Run ping test from iDRAC troubleshooting. 2. Get help from an infrastructure network administrator. For more information, see iDRAC9 RACADM CLI Guide)
	Client firewall blocking the port.	1. Check client system firewall and allow port. 2. Check LC logs.
The Metric Report that is associated with MRD is empty	Metric report definition (MRD) is not enabled	1. Set the MRD property <code>MetricReportDefinitionEnabled</code> to true. 2. Check LC logs. For more information, see Enabling Metric Report Definition Using Redfish
	MRD property "ReportActions" does not have "RedfishEvent"	Set the MRD property <code>ReportActions</code> to include <code>RedfishEvent</code> . For more information, see Configurable Properties in MRD and Enabling Metric Report Definition Using Redfish .

Topics:

- [Telemetry Error Messages](#)
- [SCP Related Possible Issues](#)

Telemetry Error Messages

Telemetry provides error messaging from the following sources:

- Dell MessageRegistry – [MessageRegistry](#) prefixed with iDRAC.
- Dell Base MessageRegistry - Redfish Specification [Base MessageRegistry](#) prefixed with Base.

The following are the commonly used Telemetry messages:

Table 38. Commonly used Telemetry messages

Message ID	MessageRegistry Location	Message
Success	DMTF BaseRegistry	Successfully Completed Request
Created	DMTF BaseRegistry	The resource has been created successfully.

Table 38. Commonly used Telemetry messages (continued)

Message ID	MessageRegistry Location	Message
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 was not found.
ResourceNotFound	DMTF BaseRegistry	The requested resource of type named " 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 was 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 that is entered for the property 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 that are associated with the current session to perform the requested operation
SYS482	Dell MessageRegistry	Unable to complete the operation because the MetricReportDefinition exists.
SYS484	Dell MessageRegistry	Unable to complete the operation because bounds are to
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

Table 38. Commonly used Telemetry messages (continued)

Message ID	MessageRegistry Location	Message
		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 iDRAC User Guides and other manuals www.dell.com/iDRACmanuals CollectionFunction must be set simultaneously. 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 was cleared.

SCP Related Possible Issues

Missing License error

If the Datacenter license is missing, then the configuration results show SYS319 message ID for CustomComponents.

SYS319 - The operation cannot be completed because either the required license is missing or expired.

```
racadm>>lclog viewconfigresult -j JID_034847857859
SeqNumber = 8049 514
Job Name = Import Configuration Operation
Name = CHANGE
DisplayValue = iDRAC.Embedded.1#CustomComponents
Name = CustomComponents OldValue = "" NewValue = "" Status = Failure MessageID = SYS319
ErrCode = 12562
```

Telemetry Not Enabled

If the Telemetry feature is not enabled, then the configuration results show a SYS460 message ID for CustomComponents.

SYS460 - Unable to perform the necessary Telemetry operation because the Telemetry feature is disabled.

```
racadm>>lclog view configresult -j JID_034845831354 SeqNumber = 8002 FQDD =  
iDRAC.Embedded.1 Job Name = Import Configuration Operation Name = CHANGE DisplayValue  
= iDRAC.Embedded.1#CustomComponents Name = CustomComponents OldValue = "" NewValue = ""  
Status = Failure MessageID = SYS460 ErrCode = 10376
```

Schema Validation Error

If the CustomComponents element is in a template that is imported to any iDRAC version prior to version 4.40.00.00, then the resulting job status is Failed with a message id of SYS047.

SYS047 - Input file for import configuration is not compliant with the configuration schema.

```
racadm>>jobqueue view -i JID_035451979234  
Output:  
JobID=JID_035451979234]  
Job Name=Configure: Import Server Configuration Profile Status=Failed Scheduled Start  
Time=[Not Applicable] Expiration Time=[Not Applicable] Actual Start Time=[Sat, 24 Oct  
2020 08:13:17] 514 Actual Completion Time=[Sat, 24 Oct 2020 08:13:18] Message=[SYS047:  
Input file for import configuration is not compliant with configuration schema.] Percent  
Complete=[100]
```

Technical support and resources

Technical support and resources:

- iDRAC Telemetry Scripts 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
- Dell Technical Support Dell.com/support
- Telemetry Reference Tools: [GitHub - dell/iDRAC-Telemetry-Reference-Tools: Reference toolset for PowerEdge telemetry metric collection and integration with analytics and visualization solutions.](#)
- Developer.Dell.com APIs [Explore APIs | Dell Technologies Developer](#)

Getting help

Topics:

- [Contacting Dell](#)

Contacting Dell

Prerequisites

 **NOTE:** If you do not have an active Internet connection, you can find contact information on your purchase invoice, packing slip, bill, or Dell product catalog.

About this task

Dell provides several online and telephone-based support and service options. Availability varies by country and product, and some services may not be available in your area. To contact Dell for sales, technical support, or customer service issues:

Steps

1. Go to **Dell.com/support**.
2. Select your support category.
3. Verify your country or region in the **Choose a Country/Region** drop-down list at the bottom of the page.
4. Select the appropriate service or support link based on your need.