Technical white paper

Manage iDRAC alerts by using the Redfish Eventing feature in YX3X and later PowerEdge servers

Abstract

In PowerEdge servers, alerts are generated and delivered to predict or foresee any actions to be done. This technical white paper enables you to configure and subscribe for alerts generated in iDRAC by using the Redfish Eventing feature.

August 2021

Revisions

Date	Description
August 2021	Initial release

Acknowledgements

Author: Venkataragavan V

Support: Mahesh Babu R and Ajay Shenoy

Others: Sheshadri PR Rao (InfoDev)

Contents

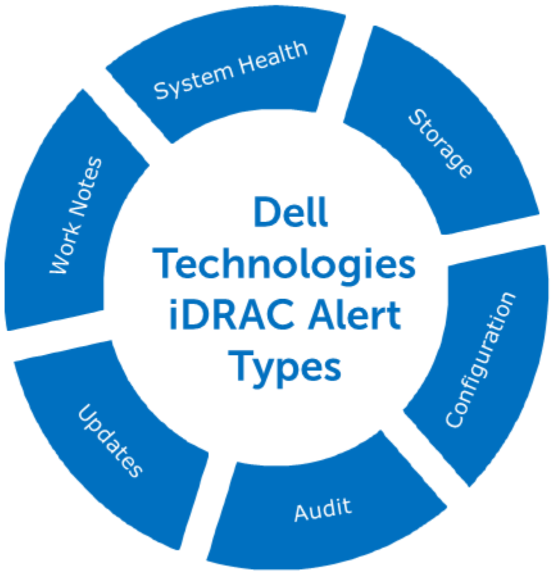
Executive summary

In the entire server lifecycle, several events may occur that administrators must be notified about to take corrective and preventive actions. Therefore, iDRAC supports the alerts-notification feature by using the Redfish Eventing feature. Alerts are streamed to client-created destinations either through Server Sent Events (SSE) or HTTP Push subscription mechanism. iDRAC8 supports Redfish Eventing for alerts by using the HTTP push style Eventing. However, iDRAC9 and later versions support both SSE and HTTP push style Eventing.

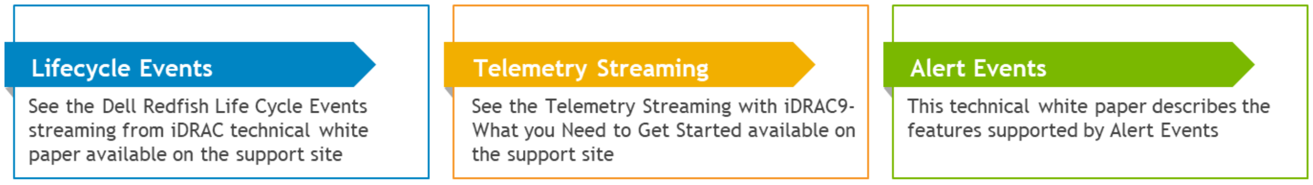
Each alert event includes necessary information such as Message, Severity, and Resolution that helps administrators quickly identify issues in the data center environment. The alerts combined with Telemetry and Lifecycle events enables administrators to analyze the failure trend or predict the future failures.

1 Overview of Redfish Eventing for iDRAC alerts

A server, in its lifetime, would go through several events such as fan failure, increase in temperature, Power Supply Unit (PSU) failure, and driver software issues. These events must be captured and sent to the customers for further actions. To do this, iDRAC provides a mechanism for notifying about alerts in different ways such as SNMP alerts, Email alerts. Similarly, Redfish Eventing mechanism is also one of the ways to send the iDRAC alerts. iDRAC alerts are broadly classified as shown in the infographics:



The Redfish Event Service supports the following event types:


Lifecycle Events
See the Dell Redfish Life Cycle Events streaming from iDRAC technical white paper available on the support site

Telemetry Streaming
See the Telemetry Streaming with iDRAC9- What you Need to Get Started available on the support site

Alert Events
This technical white paper describes the features supported by Alert Events

The Redfish Event Service provides the ability to subscribe for the alerts generated in the iDRAC by using Redfish Eventing.

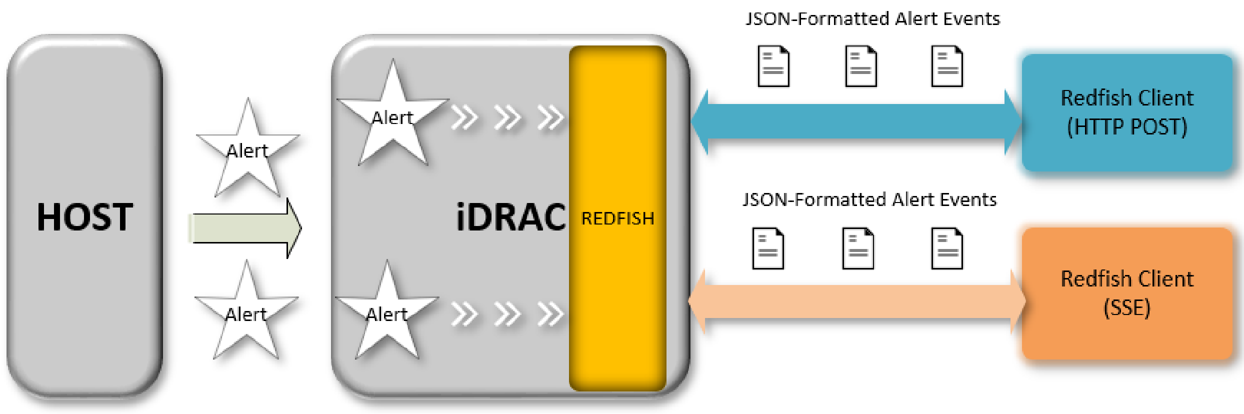

JSON-Formatted Alert Events
HOST
Alert
Alert
Alert
Alert
iDRAC
REDFISH
Redfish Client (HTTP POST)
JSON-Formatted Alert Events
Redfish Client (SSE)

Figure 1 Overview of Redfish Eventing for alerts

Redfish Event Service supports two styles of subscription.

- Push Style Eventing (HTTP POST)
- Server Sent Events (SSE)

Push style Eventing

- When the alert is generated in iDRAC, Redfish Event service uses an HTTP POST to push the alert events to the subscribed Redfish client.

SSE style Eventing

- The Redfish client starts an SSE connection to the iDRAC Redfish Event service by performing a GET operation on the SSE URI to receive the alert events generated in the iDRAC.

Note—SSE is supported from iDRAC9 v4.00.00.00 and later versions.

1.1 SNMP traps vs Redfish alerts

SNMP traps and/or Redfish alerts can be used for asynchronous event reporting. Both are supported in iDRAC.

- SNMP traps—iDRAC alerts are configured to generate SNMP traps.
- Redfish alerts—All iDRAC alerts that can currently generate SNMP traps can also be configured to generate Redfish event. Redfish events are supported for a larger spectrum of system events. In combination with Redfish Lifecycle events and Telemetry, a complete monitoring solution can be built.

- SNMP traps—SNMP clients require the latest MIB to be imported to decipher the traps.
- Redfish alerts—Redfish event are in human readable format and do not have such a restriction.

- SNMP traps—SNMP is preferred by older SNMP based clients.
- Redfish alerts—Modern REST or Redfish clients such as OpenStack Ironic can easily consume Redfish events.

- SNMP traps—The iDRAC SNMP implementation currently supports only GET and TRAP operations. SET operations are not supported.
- Redfish alerts—The Redfish stack supports GET, PATCH, POST, DELETE, and PUT operations.

- SNMP traps—SNMPv2 or SNMPv3 can be used to encrypt packets and send traps securely.
- Redfish alerts—Redfish events use HTTPs to send events securely.

- SNMP traps—SNMP supports only the push type.
- Redfish alerts—Redfish supports push type and ServerSentEvents. SSE uses single HTTPS GET Connection operation to subscribe and receive the alert events. This connection exists until either the client or server closes it.

1.2 Prerequisites

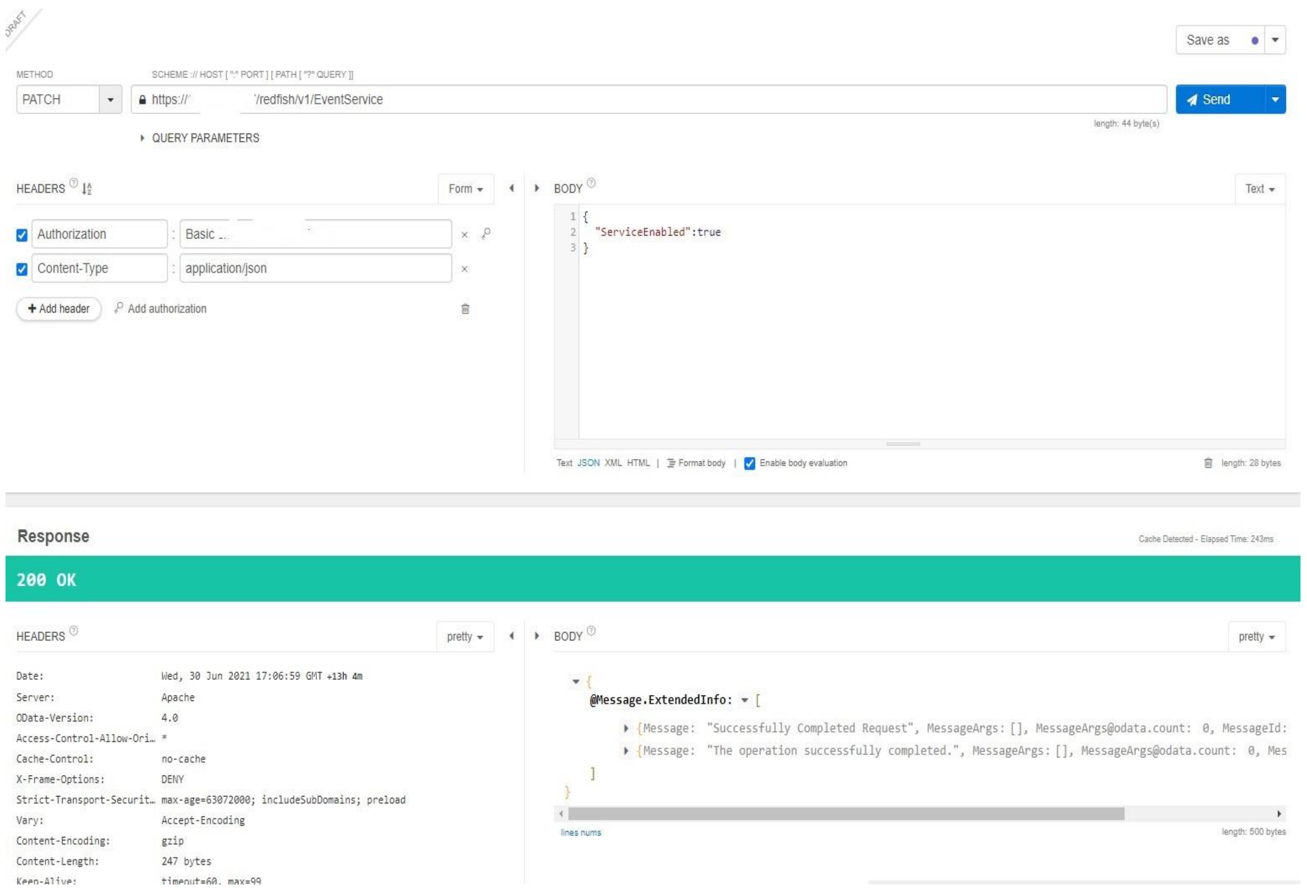
Redfish alert events can be subscribed by following event subscription as defined by DMTF redfish specification with no additional iDRAC licensing requirement from iDRAC8 2.30.30.30 and later versions.

2 Configure iDRAC alerts for Redfish events

2.1 Enable iDRAC alerts

To enable the Redfish Event Service, set the “ServiceEnabled” property in the EventService URI (`/redfish/v1/EventService`) to **Enabled**, which in turn enables the iDRAC alerts.

- **Operation**: HTTP PATCH
- **URI**: `https://<iDRAC -IP>/redfish/v1/EventService`
- **Schema**: https://redfish.dmtf.org/schemas/v1/EventService.xml


DRAFT
Save as
METHOD
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]
Send
QUERY PARAMETERS
HEADERS
Form
BODY
Text
Add header
Add authorization
Text JSON XML HTML | Format body | Enable body evaluation
length: 28 bytes
Response
Cache Detected - Elapsed Time: 243ms
pretty
lines nums
length: 500 bytes

```
PATCH   https://   /redfish/v1/EventService
length: 44 byte(s)

Authorization : Basic ...
Content-Type  : application/json

{
  "ServiceEnabled":true
}
```

```
200 OK

Date:                         Wed, 30 Jun 2021 17:06:59 GMT +13h 4m
Server:                       Apache
OData-Version:                4.0
Access-Control-Allow-Ori…     *
Cache-Control:                no-cache
X-Frame-Options:              DENY
Strict-Transport-Securit…     max-age=63072000; includeSubDomains; preload
Vary:                         Accept-Encoding
Content-Encoding:             gzip
Content-Length:               247 bytes
Keep-Alive:                   timeout=60, max=99
```

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

Figure 2 Enable iDRAC alerts for Redfish events

2.2 Configure the category of iDRAC alerts for Redfish events

iDRAC alerts can be configured through SCP (Server Configuration Profile). For more information, see the *Using Server Configuration Profiles to Deploy Operating Systems to Dell EMC PowerEdge Servers* on the support site. For information about framing payload, see http://ftp.dell.com/manuals/common/dellemc-server-config-profile-refguide.pdf.

2.3 Configure Redfish alert subscriptions

You can subscribe to Redfish Alert events by creating event subscriptions as defined by the DMTF Redfish event subscription service. iDRAC supports two styles of subscriptions—SSE and HTTP Push based. By default, this feature does not require any additional iDRAC license, but requires Redfish service and Redfish Event Service (iDRAC alerts) to be enabled to receive redfish alert events.

2.3.1 Push Style Subscription (HTTP POST)

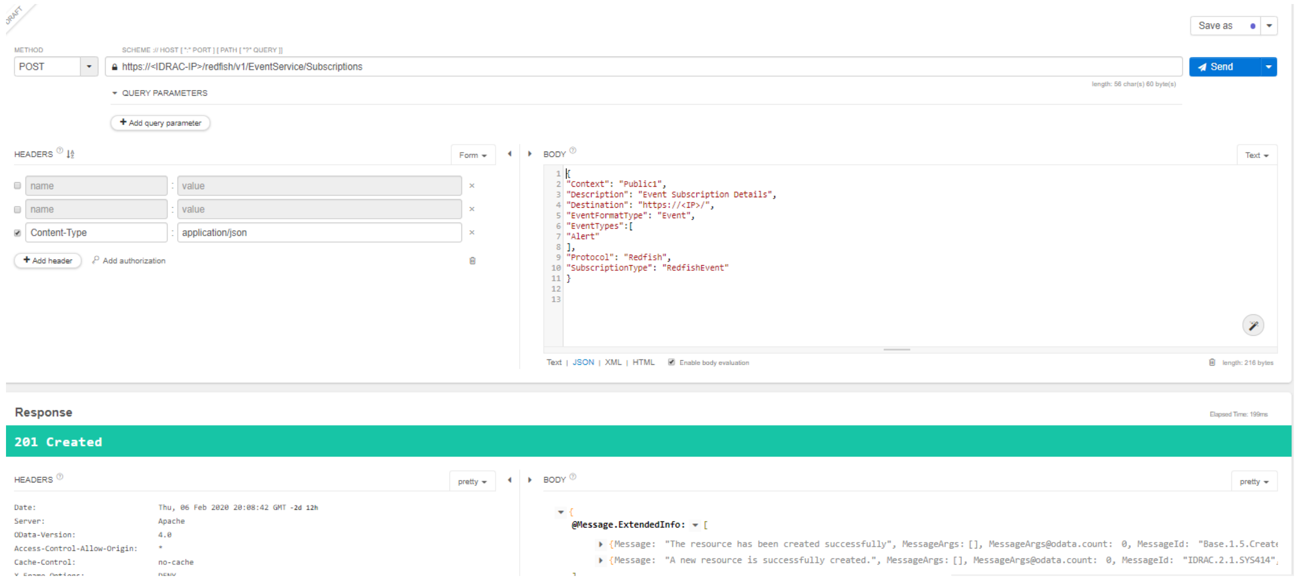
2.3.1.1 Create a Push Style subscription

Subscription for Redfish alert events, through HTTP push mechanism, is created on performing a POST operation with necessary key-value payload comprising Destination, Context, and Protocol as mandatory properties.

If you do not enter EventTypes and EventFormatType Property, by default, iDRAC Redfish event service creates alert subscription:

- **Operation**: HTTP POST
- **URI**: https://<iDRAC -IP>/redfish/v1/EventService/Subscriptions

Schema: https://redfish.dmtf.org/schemas/v1/EventDestination.xml



METHOD
POST
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]
https://<IDRAC-IP>/redfish/v1/EventService/Subscriptions

HEADERS

☐	name	value
☐	name	value
✓	Content-Type	application/json

BODY

```
{
"Context": "Public1",
"Description": "Event Subscription Details",
"Destination": "https://<IP>/",
"EventFormatType": "Event",
"EventTypes":[
"Alert"
],
"Protocol": "Redfish",
"SubscriptionType": "RedfishEvent"
}
```

Response

201 Created

HEADERS

```
Date:                          Thu, 06 Feb 2020 20:08:42 GMT -2d 12h
Server:                        Apache
OData-Version:                 4.0
Access-Control-Allow-Origin:   *
Cache-Control:                 no-cache
```

BODY

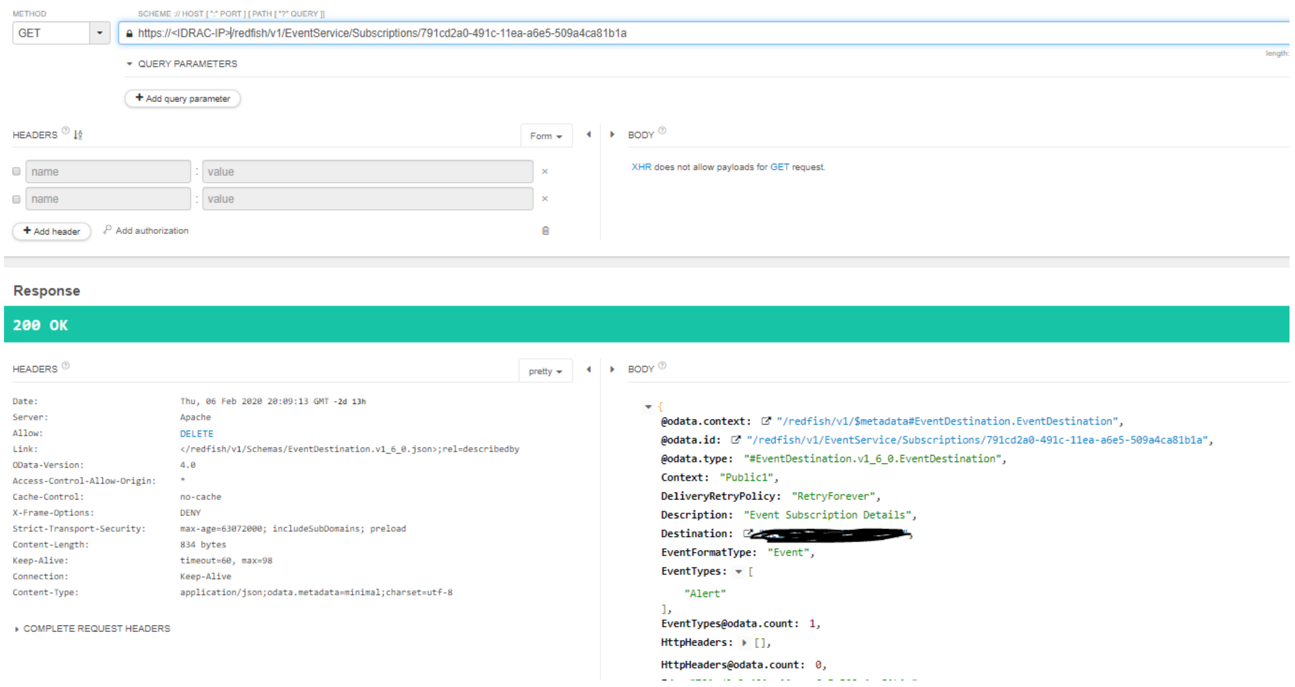
```
{
@Message.ExtendedInfo: [
  {Message:  "The resource has been created successfully", MessageArgs: [], MessageArgs@odata.count:  0, MessageId:  "Base.1.5.Create
  {Message:  "A new resource is successfully created.", MessageArgs: [], MessageArgs@odata.count:  0, MessageId:  "IDRAC.2.1.SYS414",
```

Figure 3 Create a Push style subscription

2.3.1.2 View a subscription instance

View the subscription instance you created by entering the subscription ID:

- **Operation**: HTTP GET
- **URI**: https://<iDRAC -IP>/redfish/v1/EventService/Subscriptions/<Sub-Id>
- **Schema**: https://redfish.dmtf.org/schemas/v1/EventDestination.xml



GET https://<IDRAC-IP>/redfish/v1/EventService/Subscriptions/791cd2a0-491c-11ea-a6e5-509a4ca81b1a

XHR does not allow payloads for GET request.

Response

200 OK

```
Date:                          Thu, 06 Feb 2020 20:09:13 GMT -2d 13h
Server:                        Apache
Allow:                         DELETE
Link:                          </redfish/v1/Schemas/EventDestination.v1_6_0.json>;rel=describedby
OData-Version:                 4.0
Access-Control-Allow-Origin:   *
Cache-Control:                 no-cache
X-Frame-Options:               DENY
Strict-Transport-Security:     max-age=63072000; includeSubDomains; preload
Content-Length:                834 bytes
Keep-Alive:                    timeout=60, max=98
Connection:                    Keep-Alive
Content-Type:                  application/json;odata.metadata=minimal;charset=utf-8
```

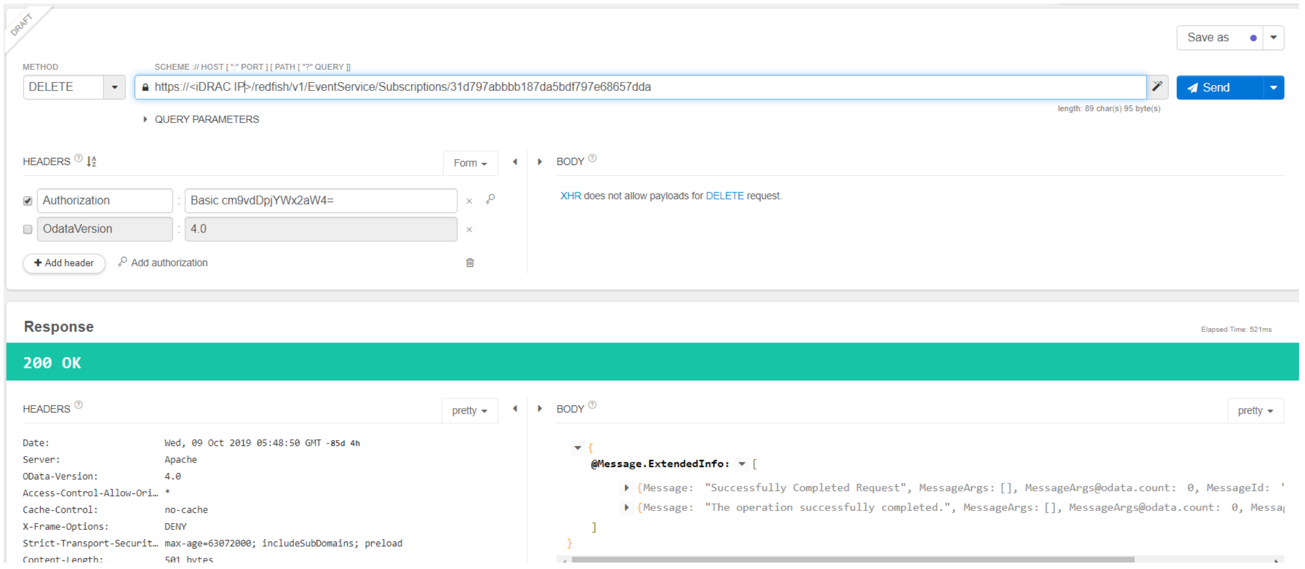
```
{
  @odata.context: "/redfish/v1/$metadata#EventDestination.EventDestination",
  @odata.id: "/redfish/v1/EventService/Subscriptions/791cd2a0-491c-11ea-a6e5-509a4ca81b1a",
  @odata.type: "#EventDestination.v1_6_0.EventDestination",
  Context: "Public1",
  DeliveryRetryPolicy: "RetryForever",
  Description: "Event Subscription Details",
  Destination: [redacted]
  EventFormatType: "Event",
  EventTypes: [
      "Alert"
  ],
  EventTypes@odata.count: 1,
  HttpHeaders: [],
  HttpHeaders@odata.count: 0,
```

Figure 4 View a subscription instance

2.3.1.3 Delete a subscription instance

Delete the subscription instance by entering the subscription ID:

- **Operation**: HTTP DELETE
- **URI**: https://<iDRAC -IP>/redfish/v1/EventService/Subscriptions/<Sub-Id>
- **Schema**: https://redfish.dmtf.org/schemas/v1/EventDestination.xml



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

Authorization	Basic cm9vdDpjYWx2aW4=
OdataVersion	4.0

XHR does not allow payloads for DELETE request.

Response

200 OK

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

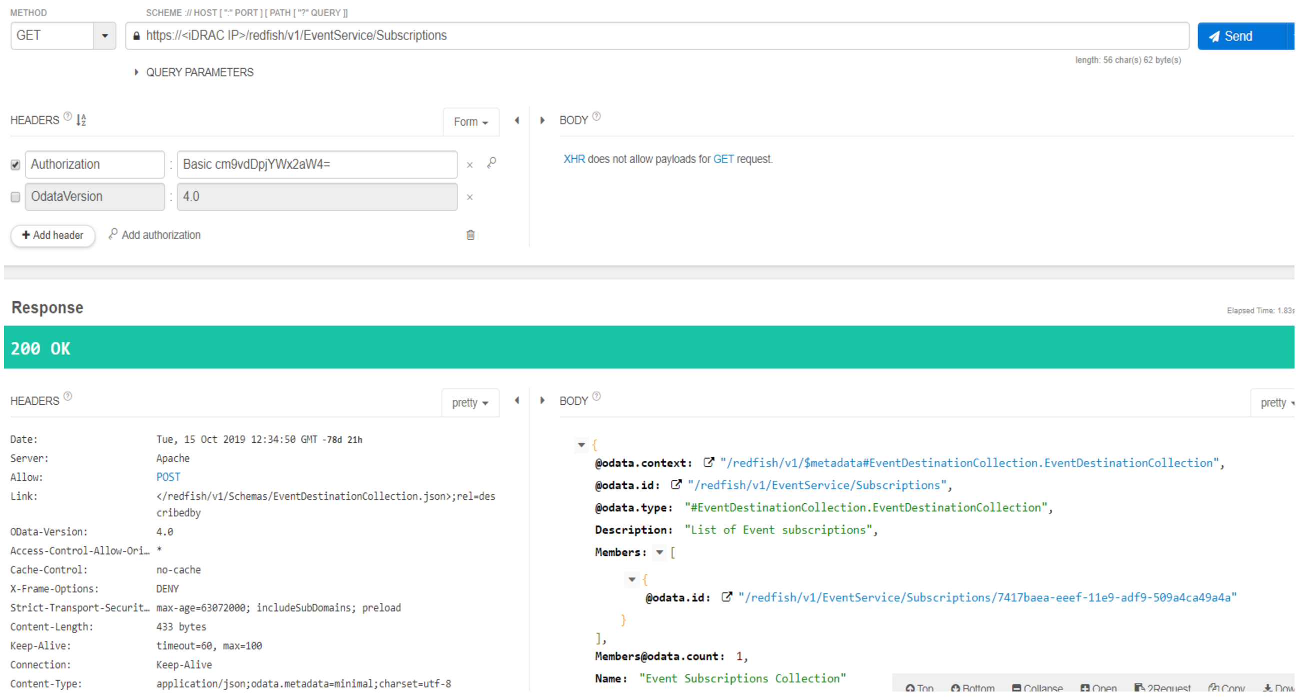
```
{
  @Message.ExtendedInfo: [
      {Message: "Successfully Completed Request", MessageArgs: [], MessageArgs@odata.count: 0, MessageId: '
      {Message: "The operation successfully completed.", MessageArgs: [], MessageArgs@odata.count: 0, Messa
  ]
}
```

Figure 5 Delete subscription instance

2.3.1.4 View the collection of subscriptions

View the created subscriptions in the subscriptions URI:

- **Operation**: HTTP GET
- **URI**: https://<iDRAC -IP>/redfish/v1/EventService/Subscriptions
- **Schema**: https://redfish.dmtf.org/schemas/v1/EventDestinationCollection.xml



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

Authorization	Basic cm9vdDpjYWx2aW4=
OdataVersion	4.0

XHR does not allow payloads for GET request.

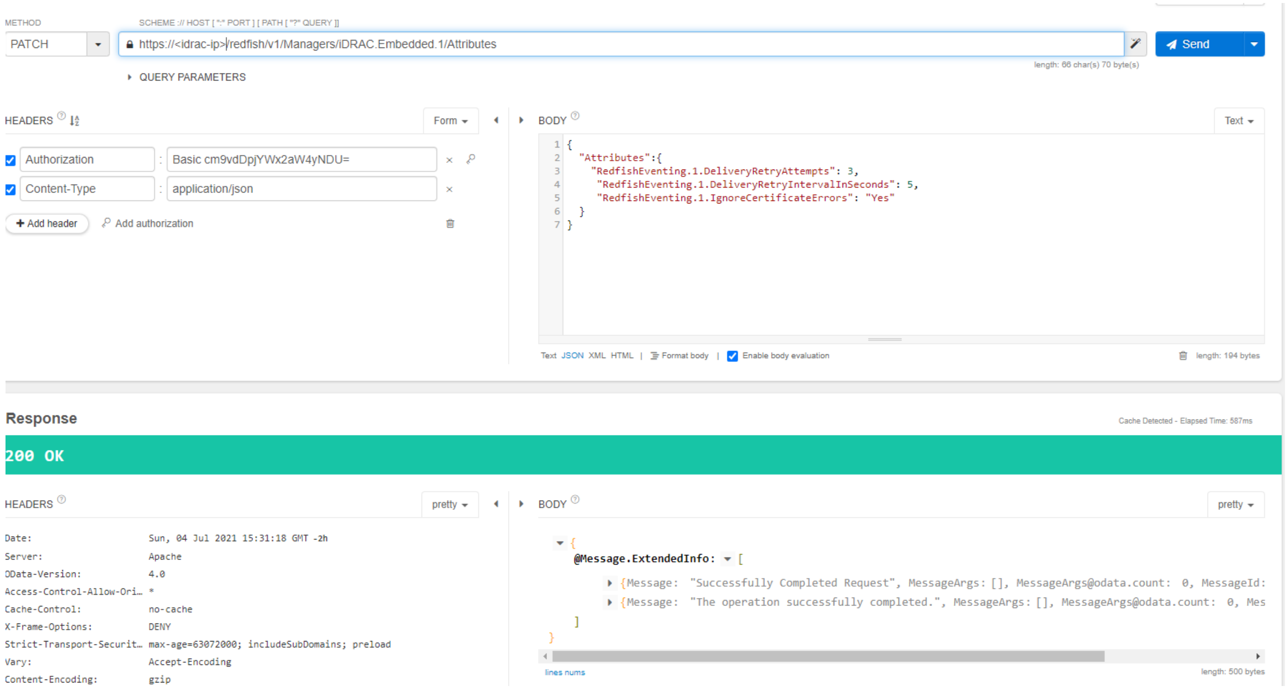
200 OK

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

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

Figure 6 View subscriptions

2.3.1.5 Configure delivery of alert events for HTTP Push style subscription


METHOD
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]
Send
QUERY PARAMETERS
HEADERS
Form
BODY
Text
Add header
Add authorization
Text JSON XML HTML | Format body | Enable body evaluation
Response
pretty
lines nums

METHOD: PATCH

https://<idrac-ip>/redfish/v1/Managers/iDRAC.Embedded.1/Attributes

length: 66 char(s) 70 byte(s)

HEADERS

Authorization	Basic cm9vdDpjYWx2aW4yNDU=
Content-Type	application/json

BODY

```
{
  "Attributes":{
    "RedfishEventing.1.DeliveryRetryAttempts": 3,
     "RedfishEventing.1.DeliveryRetryIntervalInSeconds": 5,
     "RedfishEventing.1.IgnoreCertificateErrors": "Yes"
  }
}
```

length: 194 bytes

Response

Cache Detected - Elapsed Time: 587ms

200 OK

HEADERS

```
Date:                       Sun, 04 Jul 2021 15:31:18 GMT -2h
Server:                     Apache
OData-Version:              4.0
Access-Control-Allow-Ori…   *
Cache-Control:              no-cache
X-Frame-Options:            DENY
Strict-Transport-Securit…   max-age=63072000; includeSubDomains; preload
Vary:                       Accept-Encoding
Content-Encoding:           gzip
```

BODY

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

length: 500 bytes

Figure 7 Configure delivery of alert events for HTTP Push style subscription

Redfish Eventing iDRAC Attributes:

RedfishEventing.1.DeliveryRetryAttempts

The number of times that the POST of an event to a destination is retried before the EventService goes for another subscription

RedfishEventing.1.DeliveryRetryIntervalInSeconds

The interval, in seconds, between retry attempts for sending any event to a destination.

RedfishEventing.1.IgnoreCertificateErrors

If yes, the certification validation will not happen over HTTPS.

2.3.1.6 Redfish event listener

The Redfish Event Listener is a lightweight HTTP(S) server which is used to receive the events. The configured HTTPS endpoint must be entered in the Destination property while creating the HTTP Push Style Subscription. See https://github.com/DMTF/Redfish-Event-Listener.

2.3.2 Redfish SSE subscription

Server Sent Events (SSE) is another method of delivering the events in Redfish Eventing. As defined by the Web Hypertext Application Technology Working Group, SSE enables a client to start a connection with a web service and the web service can continuously push data to the client as required.

Clients can subscribe to Redfish Alert events with subscription created over an SSE channel by performing GET on web terminals supporting SSE Protocol.

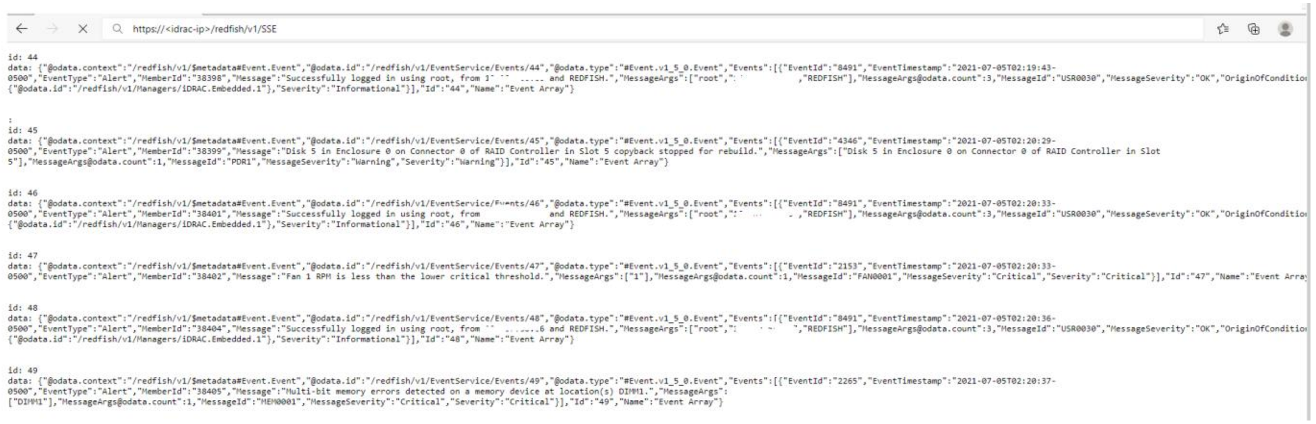
Redfish URI for SSE Alert Event streaming

1. *https://<iDRAC IP>/redfish/v1/SSE (or)*
2. *https://<iDRAC IP>/redfish/v1/SSE?$filter=EventFormatType eq Event*

Event service on receiving a "GET" request from the client makes an entry of the SSE subscription in the subscription collection and starts streaming the alert events generated by iDRAC. The ID field of SSE used must be same as the unique identifier of the event and the data field of SSE contains the Event Payload.

The SSE Connection can be ended by either Client or iDRAC event service. On cases when there is no data being sent to client for more than one hour, the connection would be ended from Event Service endpoint. When the connection is ended, the subscription entry will be removed from the subscription collection.

SSE Client Output for Alert Events



```
id: 44
data: {"@odata.context":"/redfish/v1/$metadata#Event.Event","@odata.id":"/redfish/v1/EventService/Events/44","@odata.type":"#Event.v1_5_0.Event","Events":[{"EventId":"8491","EventTimestamp":"2021-07-05T02:19:43-
0500","EventType":"Alert","MemberId":"38398","Message":"Successfully logged in using root, from [redacted] and REDFISH.","MessageArgs":["root","[redacted]","REDFISH"],"MessageArgs@odata.count":3,"MessageId":"USR0030","MessageSeverity":"OK","OriginOfConditio
{"@odata.id":"/redfish/v1/Managers/iDRAC.Embedded.1"},"Severity":"Informational"}],"Id":"44","Name":"Event Array"}

:
id: 45
data: {"@odata.context":"/redfish/v1/$metadata#Event.Event","@odata.id":"/redfish/v1/EventService/Events/45","@odata.type":"#Event.v1_5_0.Event","Events":[{"EventId":"4346","EventTimestamp":"2021-07-05T02:20:29-
0500","EventType":"Alert","MemberId":"38399","Message":"Disk 5 in Enclosure 0 on Connector 0 of RAID Controller in Slot 5 copyback stopped for rebuild.","MessageArgs":["Disk 5 in Enclosure 0 on Connector 0 of RAID Controller in Slot
5"],"MessageArgs@odata.count":1,"MessageId":"PDR1","MessageSeverity":"Warning","Severity":"Warning"}],"Id":"45","Name":"Event Array"}

id: 46
data: {"@odata.context":"/redfish/v1/$metadata#Event.Event","@odata.id":"/redfish/v1/EventService/Events/46","@odata.type":"#Event.v1_5_0.Event","Events":[{"EventId":"8491","EventTimestamp":"2021-07-05T02:20:33-
0500","EventType":"Alert","MemberId":"38401","Message":"Successfully logged in using root, from [redacted] and REDFISH.","MessageArgs":["root","[redacted]","REDFISH"],"MessageArgs@odata.count":3,"MessageId":"USR0030","MessageSeverity":"OK","OriginOfConditio
{"@odata.id":"/redfish/v1/Managers/iDRAC.Embedded.1"},"Severity":"Informational"}],"Id":"46","Name":"Event Array"}

id: 47
data: {"@odata.context":"/redfish/v1/$metadata#Event.Event","@odata.id":"/redfish/v1/EventService/Events/47","@odata.type":"#Event.v1_5_0.Event","Events":[{"EventId":"2153","EventTimestamp":"2021-07-05T02:20:33-
0500","EventType":"Alert","MemberId":"38402","Message":"Fan 1 RPM is less than the lower critical threshold.","MessageArgs":["1"],"MessageArgs@odata.count":1,"MessageId":"FAN0001","MessageSeverity":"Critical","Severity":"Critical"}],"Id":"47","Name":"Event Arra

id: 48
data: {"@odata.context":"/redfish/v1/$metadata#Event.Event","@odata.id":"/redfish/v1/EventService/Events/48","@odata.type":"#Event.v1_5_0.Event","Events":[{"EventId":"8491","EventTimestamp":"2021-07-05T02:20:36-
0500","EventType":"Alert","MemberId":"38404","Message":"Successfully logged in using root, from [redacted]6 and REDFISH.","MessageArgs":["root","[redacted]","REDFISH"],"MessageArgs@odata.count":3,"MessageId":"USR0030","MessageSeverity":"OK","OriginOfConditio
{"@odata.id":"/redfish/v1/Managers/iDRAC.Embedded.1"},"Severity":"Informational"}],"Id":"48","Name":"Event Array"}

id: 49
data: {"@odata.context":"/redfish/v1/$metadata#Event.Event","@odata.id":"/redfish/v1/EventService/Events/49","@odata.type":"#Event.v1_5_0.Event","Events":[{"EventId":"2265","EventTimestamp":"2021-07-05T02:20:37-
0500","EventType":"Alert","MemberId":"38405","Message":"Multi-bit memory errors detected on a memory device at location(s) DIMM1.","MessageArgs":
["DIMM1"],"MessageArgs@odata.count":1,"MessageId":"MEM0001","MessageSeverity":"Critical","Severity":"Critical"}],"Id":"49","Name":"Event Array"}
```

2.3.2.1 View and delete SSE subscription

Both SSE and Push Style Subscriptions will be listed here: `URI: https://<iDRAC - IP>/redfish/v1/EventService/Subscriptions`

Also see the following sections:

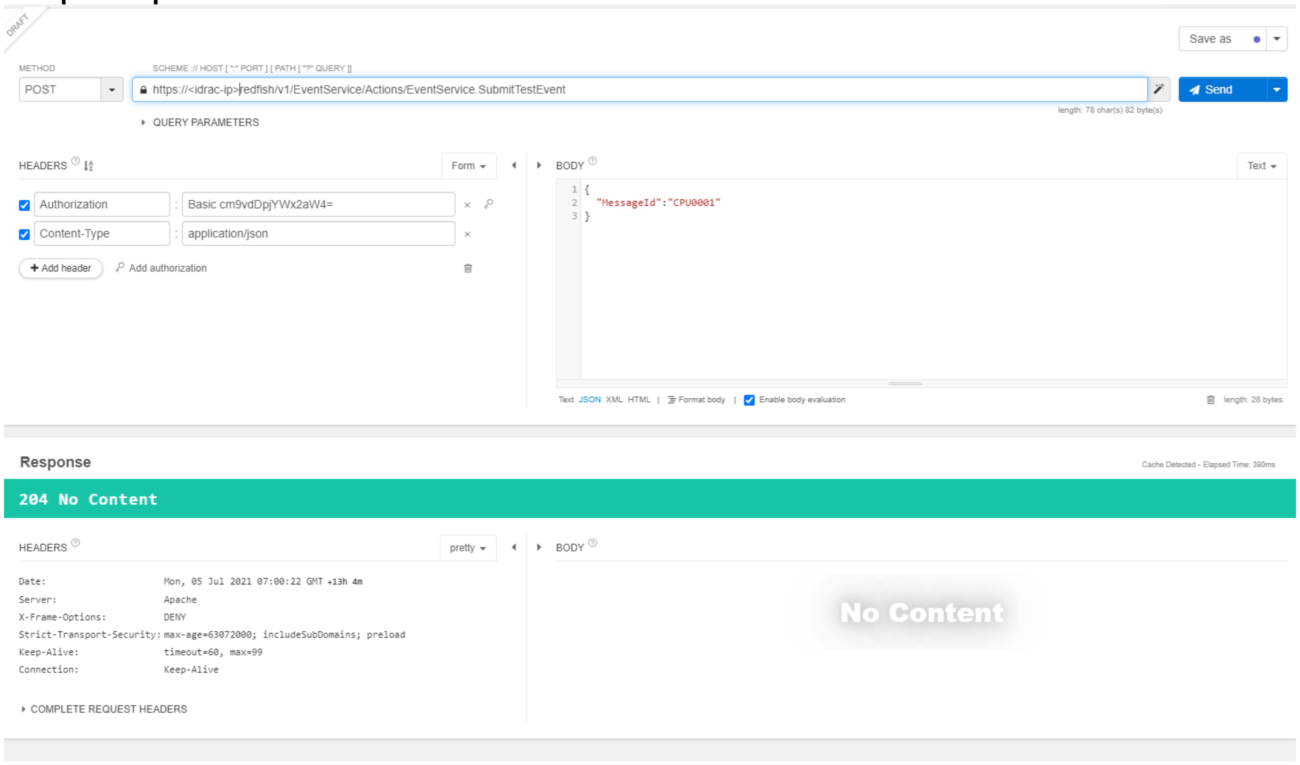
- Delete subscription
- View subscriptions collection

2.4 Test event

Client can use the SubmitTestEvent action to generate the test events for testing if the destination is properly configured and receiving the events.

Schema: https://redfish.dmtf.org/schemas/v1/EventService.xml

Sample Output


DRAFT
Save as
METHOD
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]
Send
QUERY PARAMETERS
Form
Text
Add header
Add authorization
Text JSON XML HTML | Format body | Enable body evaluation
pretty
COMPLETE REQUEST HEADERS

```
POST https://<idrac-ip>redfish/v1/EventService/Actions/EventService.SubmitTestEvent
length: 78 char(s) 82 byte(s)
```

HEADERS

Authorization	Basic cm9vdDpjYWx2aW4=
Content-Type	application/json

BODY

```
{
  "MessageId":"CPU0001"
}
```

length: 28 bytes

Response

Cache Detected - Elapsed Time: 390ms

```
204 No Content
```

HEADERS

```
Date:                      Mon, 05 Jul 2021 07:00:22 GMT +13h 4m
Server:                    Apache
X-Frame-Options:           DENY
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Keep-Alive:                timeout=60, max=99
Connection:                Keep-Alive
```

BODY

No Content

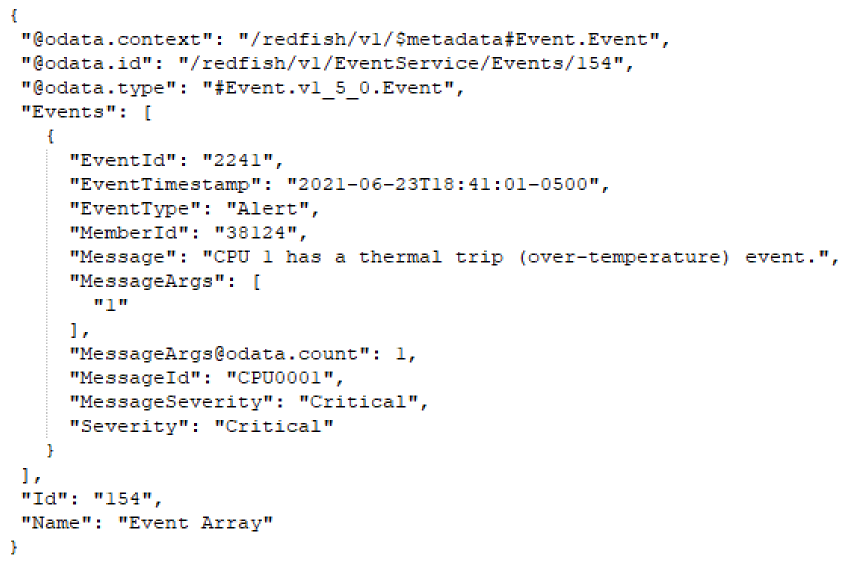
Figure 8 Test if destination is configured and receiving events

2.5 Redfish alert event format

The Redfish alert event which is delivered to both SSE and HTTP Push Style uses the following Event Schema:

Schema: https://redfish.dmtf.org/schemas/v1/Event.xml

Sample Output



```
{
 "@odata.context": "/redfish/v1/$metadata#Event.Event",
 "@odata.id": "/redfish/v1/EventService/Events/154",
 "@odata.type": "#Event.v1_5_0.Event",
 "Events": [
   {
     "EventId": "2241",
     "EventTimestamp": "2021-06-23T18:41:01-0500",
     "EventType": "Alert",
     "MemberId": "38124",
     "Message": "CPU 1 has a thermal trip (over-temperature) event.",
     "MessageArgs": [
       "1"
     ],
     "MessageArgs@odata.count": 1,
     "MessageId": "CPU0001",
     "MessageSeverity": "Critical",
     "Severity": "Critical"
   }
 ],
 "Id": "154",
 "Name": "Event Array"
}
```

2.6 Troubleshooting tips

Problem	**Possible Solutions**
Redfish client not getting Alerts in Push Style	1. Ensure that the Global alert is enabled. 2. Ensure that the Subscription Destination is configured properly to receive alert events.
Redfish client not getting Alerts in SSE Style	1. Ensure that the Global alert is enabled. 2. Ensure that the SSE connection is intact with iDRAC.

3 Best practices

The most suitable way to configure iDRAC Redfish Alerts Eventing is by using Server Configuration Profile (SCP). After an SCP file is created, the same file can be applied to multiple servers.

4 Redfish Events: Alerts / Telemetry / Life cycle events

Feature	Alerts	Telemetry	Life cycle events
System Anomalies	✓	✗	✗
Metrics – Time series data	✗	✓	✗
Resource - Updates/Actions	✗	*✓	✓
Triggers	✗	✓	✗
Supports HTTP Push and SSE Redfish clients	✓	✓	✓
Supported iDRAC Licenses	Express, Enterprise, Datacenter	OMEA Advanced, Datacenter	Express, Enterprise, Datacenter

A References

Telemetry

- Telemetry Streaming with iDRAC9 — What you Need to Get Started (dell.com)

Redfish legacy events (Alerts)

- Implementation of the DMTF Redfish API on Dell EMC PowerEdge Servers
- https://www.dell.com/support/manuals/en-us/idrac7-8-lifecycle-controller-v2.40.40.40/redfish%202.40.40.40/eventing?guid=guid-ab574b6d-b473-4e10-9916-5d4f7e395e0d&lang=en-us

Redfish Event Listener

- https://github.com/DMTF/Redfish-Event-Listener