

Simulate an iDRAC Redfish Environment Using Redfish Mockup Server

Abstract

Leveraging Redfish mockup server on GitHub enables you to simulate an Integrated Dell Remote Access Controller (iDRAC) Redfish environment without the need of access to a physical server or iDRAC.

February 2024

Revisions

Date	Description
February 2024	Initial release

Acknowledgments

Author: Texas Roemer

Support: Nikita Joshi, Rich Schnur, and Ajay Shenoy

Other: Samiksha Aggarwal (Content Engineering)

The information in this publication is provided “as is.” Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

This document may contain certain words that are not consistent with Dell's current language guidelines. Dell plans to update the document over subsequent future releases to revise these words accordingly.

This document may contain language from third party content that is not under Dell's control and is not consistent with Dell's current guidelines for Dell's own content. When such third party content is updated by the relevant third parties, this document will be revised accordingly.

Copyright © 2024 Dell Inc. or its subsidiaries. All Rights Reserved. Dell Technologies, Dell, and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be trademarks of their respective owners. [2/28/2024] [Technical White Paper] [613]

Contents

Revisions.....2

Acknowledgments.....2

Contents.....3

Executive Summary4

1 iDRAC Redfish Mockup Client.....5

1.1 Leverage iDRAC Mockup GitHub repository5

1.2 Create iDRAC Mockup Client5

2 Redfish Mockup Server7

2.1 Start Redfish Mockup Server.....7

2.2 Run Simulated iDRAC Redfish Calls.....7

3 Simulate streaming iDRAC Redfish events workflow29

3.1 Configure Redfish Event Listener.....29

3.2 Create subscription.....30

3.3 Get subscription information.....30

3.4 Start Redfish Event Listener.....32

3.5 Submit test event.....33

3.6 Confirm Redfish Event Listener Receives Event.....33

4 Limitations.....35

5 Helpful Documentation Links.....36

Executive Summary

Redfish mockup server is an effective method to simulate Redfish iDRAC environment without using a physical server or iDRAC. Leveraging Redfish mockup server can solve the following issues:

- Do not have access to a physical server or iDRAC.
- Do have access to a server and you must test Redfish `PATCH` or `POST` calls without being destructive.
- Train many team members on Redfish without the necessity of physical servers or iDRAC.
- Do not have iDRAC user privileges to make Redfish calls.

1 iDRAC Redfish Mockup Client

To run the Redfish mockup server, you must have a Redfish mockup client from a real iDRAC. You can either leverage the mockup client from iDRAC Redfish GitHub site or you can create your own.

1.1 Leverage iDRAC Mockup GitHub repository

Multiple iDRAC Redfish mockup clients are available on iDRAC Redfish GitHub Repository:

<https://github.com/dell/iDRAC-Redfish-Scripting/tree/master/iDRAC%20Redfish%20Mockup%20Clients>

Extract the mockup file to view the content that can be leveraged by the Redfish mockup server.

1.2 Create iDRAC Mockup Client

To create your own iDRAC mockup client, you need the Redfish Mockup Creator available on GitHub:

<https://github.com/DMTF/Redfish-Mockup-Creator>

Access to a iDRAC (iDRAC IP and user credentials) is required to run the Redfish Mockup Creator Python script.

Here is an example to run the Redfish Mockup Creator Python script at the Windows Command Line Interface (CLI):

```
python redfishMockupCreate.py -u root -p calvin -r 192.168.0.120 -D
C:\Python310\iDRAC_Redfish_mockup_client -S
```

Note—Script may take 5–10 minutes to complete based on server configuration content and Lifecycle Controller log data size.

Snippet example of running Python script to create the Redfish mockup client:

```
C:\Python310>python redfishMockupCreate.py -u root -p calvin -r 192.168.0.120 -D
C:\Python310\iDRAC_mockup_client -S
Redfish Mockup Creator, Version 1.2.0
Address: https://192.168.0.120
Full Output Path: C:\Python310\iDRAC_mockup_client
Description:
Starting mockup creation...
Getting /redfish...
Getting /redfish/v1/odata...
Getting /redfish/v1/$metadata...
Getting /redfish/v1/Schemas/AccelerationFunction_v1.xml...
Getting /redfish/v1/Schemas/RedfishExtensions_v1.xml...
Getting /redfish/v1/Schemas/Settings_v1.xml...
Getting /redfish/v1/Schemas/Resource_v1.xml...
Getting /redfish/v1/Schemas/LogEntry_v1.xml...
Getting /redfish/v1/Schemas/Event_v1.xml...
Getting /redfish/v1/Schemas/Message_v1.xml...
Getting /redfish/v1/Schemas/CollectionCapabilities_v1.xml...
Getting /redfish/v1/Schemas/Certificate_v1.xml...
Getting /redfish/v1/Schemas/ResourceBlock_v1.xml...
```

```

Getting /redfish/v1/Schemas/Chassis_v1.xml...
Getting /redfish/v1/Schemas/Thermal_v1.xml...
Getting /redfish/v1/Schemas/PhysicalContext_v1.xml...
Getting /redfish/v1/Schemas/Redundancy_v1.xml...
Getting /redfish/v1/Schemas/Assembly_v1.xml...
Getting /redfish/v1/Schemas/Power_v1.xml...
Getting /redfish/v1/Schemas/Manager_v1.xml...
Getting /redfish/v1/Schemas/ComputerSystem_v1.xml...
Getting /redfish/v1/Schemas/LogServiceCollection_v1.xml...
Getting /redfish/v1/Schemas/LogService_v1.xml...
Getting /redfish/v1/Schemas/LogEntryCollection_v1.xml...
Getting /redfish/v1/Schemas/EthernetInterfaceCollection_v1.xml...
Getting /redfish/v1/Schemas/EthernetInterface_v1.xml...
Getting /redfish/v1/Schemas/IPAddresses_v1.xml...
Getting /redfish/v1/Schemas/VlanNetworkInterface_v1.xml...
Getting /redfish/v1/Schemas/VlanNetworkInterfaceCollection_v1.xml...
Getting /redfish/v1/Schemas/Endpoint_v1.xml...
Getting /redfish/v1/Schemas/Port_v1.xml...
Getting /redfish/v1/Schemas/Switch_v1.xml...
Getting /redfish/v1/Schemas/PortCollection_v1.xml...
Getting /redfish/v1/Schemas/Protocol_v1.xml...
Getting /redfish/v1/Schemas/PCIeDevice_v1.xml...
Getting /redfish/v1/Schemas/PCIeFunction_v1.xml...

```

Redfish mockup client contents example:

Name	Date modified	Type	Size
Smetadata	1/30/2024 12:41 PM	File folder	
AccountService	1/30/2024 12:42 PM	File folder	
CertificateService	1/30/2024 12:42 PM	File folder	
Chassis	1/30/2024 12:42 PM	File folder	
ComponentIntegrity	1/30/2024 12:44 PM	File folder	
EventService	1/30/2024 12:44 PM	File folder	
Fabrics	1/30/2024 12:44 PM	File folder	
JobService	1/30/2024 12:44 PM	File folder	
JsonSchemas	1/30/2024 12:46 PM	File folder	
LicenseService	1/30/2024 12:46 PM	File folder	
Managers	1/30/2024 12:46 PM	File folder	
odata	1/30/2024 12:41 PM	File folder	
Registries	1/30/2024 12:46 PM	File folder	
Schemas	1/30/2024 12:46 PM	File folder	
SessionService	1/30/2024 12:46 PM	File folder	
Systems	1/30/2024 12:46 PM	File folder	
TaskService	1/30/2024 12:46 PM	File folder	
TelemetryService	1/30/2024 12:47 PM	File folder	
UpdateService	1/30/2024 12:47 PM	File folder	
index.json	1/30/2024 12:41 PM	JSON Source File	3 KB

Figure 1 Redfish mockup client content

2 Redfish Mockup Server

The Redfish mockup server serves the Redfish requests for Redfish mockup client. After the Redfish mockup server is initiated, run the Redfish calls to simulate real time behavior.

Note—Redfish mockup server runs at either a specified IP address, port, or the default localhost IP address/port, 127.0.0.1:8000.

2.1 Start Redfish Mockup Server

To run any Redfish simulation calls, you must start the Redfish mockup server leveraging the Python script from GitHub:

<https://github.com/DMTF/Redfish-Mockup-Server>

Note—The `grequests` module is required to run the Redfish mockup server. You can install this module using the `pip3` file.

An example to start the Redfish mockup server which directs to the Redfish mockup client directory:

```
C:\Python310>python redfishMockupServer.py -D C:\Python310\iDRAC_mockup_client -X
Redfish Mockup Server, version 1.2.4
Hostname: 127.0.0.1
Port: 8000
Mockup directory path specified: C:\Python310\iDRAC_mockup_client
Response time: 0 seconds
Serving Mockup in absolute path: C:\Python310\iDRAC_mockup_client
Serving Redfish mockup on port: 8000
running Server...
```

Note—Do not end the CLI session, which is running the Redfish mockup server. This action will end the Redfish mockup server session.

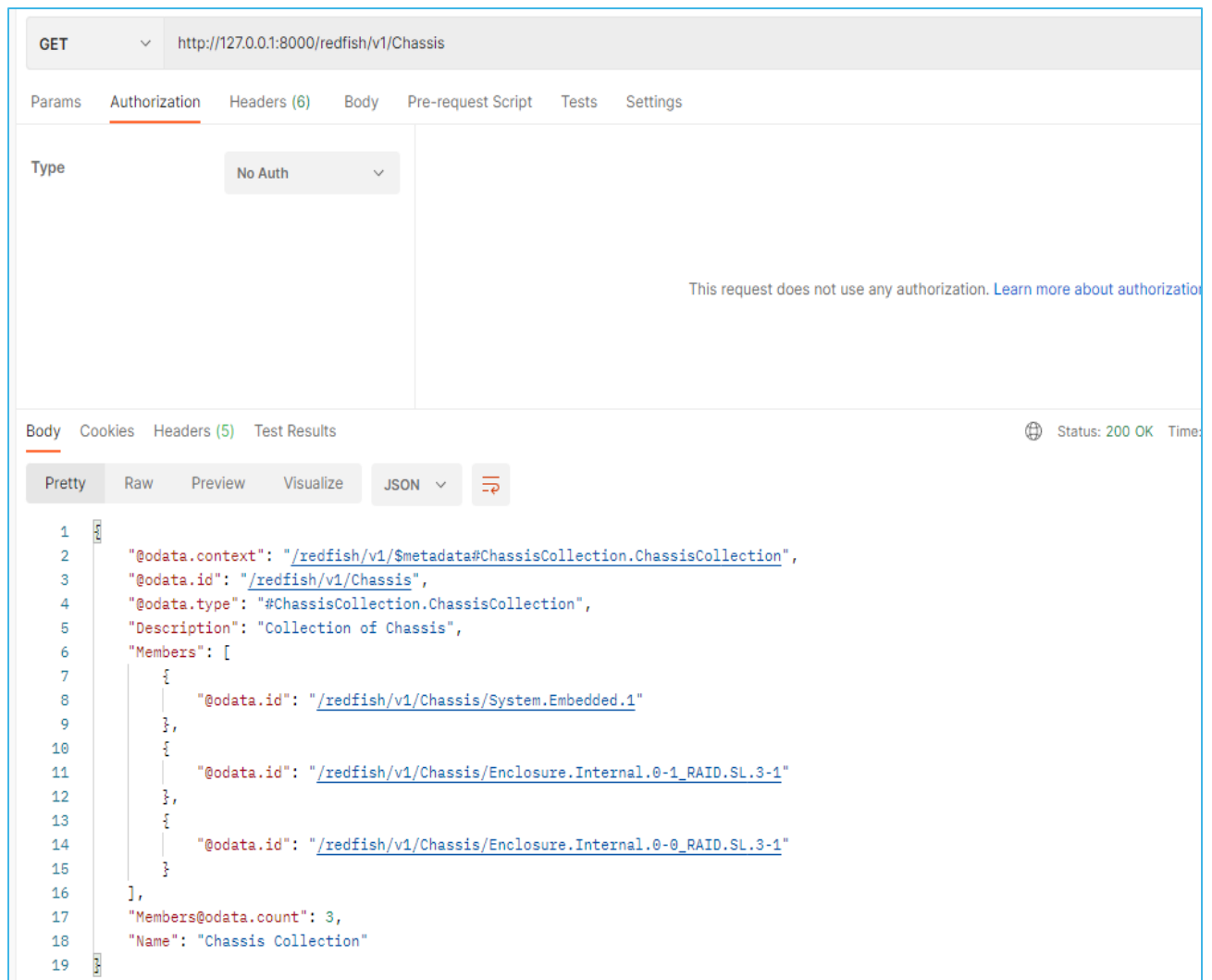
2.2 Run Simulated iDRAC Redfish Calls

When the Redfish mockup server is running, start the Redfish calls operation using any supported method (for example: Python, Curl, Postman, or PowerShell). To run Redfish calls, use the specified IP address/port or the default localhost IP address/port.

For more examples of Redfish workflow, see [Redfish FAQ Whitepaper](#).

2.2.1 GET Request for Chassis schema

The following example indicates the `GET` request for Chassis schema using the Postman and mockup server logs:



GET ▼ http://127.0.0.1:8000/redfish/v1/Chassis

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Type No Auth ▼

This request does not use any authorization. [Learn more about authorization](#)

Body Cookies Headers (5) Test Results ⊕ Status: 200 OK Time:

Pretty Raw Preview Visualize JSON ⌵

```

1  {
2    "@odata.context": "/redfish/v1/$metadata#ChassisCollection.ChassisCollection",
3    "@odata.id": "/redfish/v1/Chassis",
4    "@odata.type": "#ChassisCollection.ChassisCollection",
5    "Description": "Collection of Chassis",
6    "Members": [
7      {
8        "@odata.id": "/redfish/v1/Chassis/System.Embedded.1"
9      },
10     {
11       "@odata.id": "/redfish/v1/Chassis/Enclosure.Internal.0-1_RAID.SL.3-1"
12     },
13     {
14       "@odata.id": "/redfish/v1/Chassis/Enclosure.Internal.0-0_RAID.SL.3-1"
15     }
16   ],
17   "Members@odata.count": 3,
18   "Name": "Chassis Collection"
19 }

```

Figure 2 GET request for Chassis schema

```

C:\Python310>python redfishMockupServer.py -D C:\Python310\iDRAC_mockup_client -X
Redfish Mockup Server, version 1.2.4
Hostname: 127.0.0.1
Port: 8000
Mockup directory path specified: C:\Python310\iDRAC_mockup_client
Response time: 0 seconds
Serving Mockup in absolute path: C:\Python310\iDRAC_mockup_client
Serving Redfish mockup on port: 8000
running Server`
('GET', '/redfish/v1/Chassis')
  GET: Headers: User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 90a5c701-ae39-4e05-8fe5-369691c342ff
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

```



```

127.0.0.1 - - [30/Jan/2024 13:36:1"] "GET /redfish/v1/Chassis HTTP/1.1" 200
('GET', '/redfish/v1/Chassis')
GET: Headers: User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: c890fba0-756e-463b-95f2-9331d4fd36fe
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

```

2.2.2 GET Request for UpdateService schema

The following example indicates the GET request for UpdateService schema using the \$expand query parameter and mockup server logs:

The screenshot shows a Postman interface for a GET request to `http://127.0.0.1:8000/redfish/v1/UpdateService/FirmwareInventory?$expand=*(($levels=1))`. The request is configured with 'No Auth' and returns a 200 OK status. The response body is displayed in JSON format, showing a collection of firmware inventory items with detailed OData metadata.

```

1  {
2    "@odata.context": "/redfish/v1/$metadata#SoftwareInventoryCollection.SoftwareInventoryCollection",
3    "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory",
4    "@odata.type": "#SoftwareInventoryCollection.SoftwareInventoryCollection",
5    "Description": "Collection of Firmware Inventory",
6    "Members": [
7      {
8        "@odata.context": "/redfish/v1/$metadata#SoftwareInventory.SoftwareInventory",
9        "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/Installed-0-16.35.30.06__NIC.Integrated.1-1-1",
10       "@odata.type": "#SoftwareInventory.v1_9_0.SoftwareInventory",
11       "Description": "Represents Firmware Inventory",
12       "Id": "Installed-0-16.35.30.06__NIC.Integrated.1-1-1",
13       "Name": "Mellanox ConnectX-5 EN 25GbE Dual-port SFP28 Adapter - B8:CE:F6:96:26:0E",
14       "Oem": {
15         "Dell": {
16           "@odata.type": "#DellOem.v1_3_0.DellOemResources",
17           "DellSoftwareInventory": {
18             "@odata.context": "/redfish/v1/$metadata#DellSoftwareInventory.DellSoftwareInventory",
19             "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/Installed-0-16.35.30.06__NIC.Integrated.1-1-1/Oem/Dell/DellSoftw
DCIM_INSTALLED_0x23_781__NIC.Integrated.1-1-1",

```

Figure 3 GET request on UpdateService schema

```

127.0.0.1 - - [06/Feb/2024 12:54:56] "GET
/redfish/v1/UpdateService/FirmwareInventory HTTP/1.1" 200 -
('GET', '/redfish/v1/UpdateService/FirmwareInventory?$expand=*(($levels=1)')
GET: Headers: User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 6e0a137d-f117-47f8-92b8-ae7f2c1eed1
Host: 127.0.0.1:8000

```

Accept-Encoding: gzip, deflate, br
 Connection: keep-alive

2.2.3 PATCH Request for BootSourceOverrideTarget property

The following example indicates:

- A **PATCH** request to change the `BootSourceOverrideTarget` value from `None` to `PXE`.
- A **GET** request to view updated property value (following example also shows the mockup server logs):

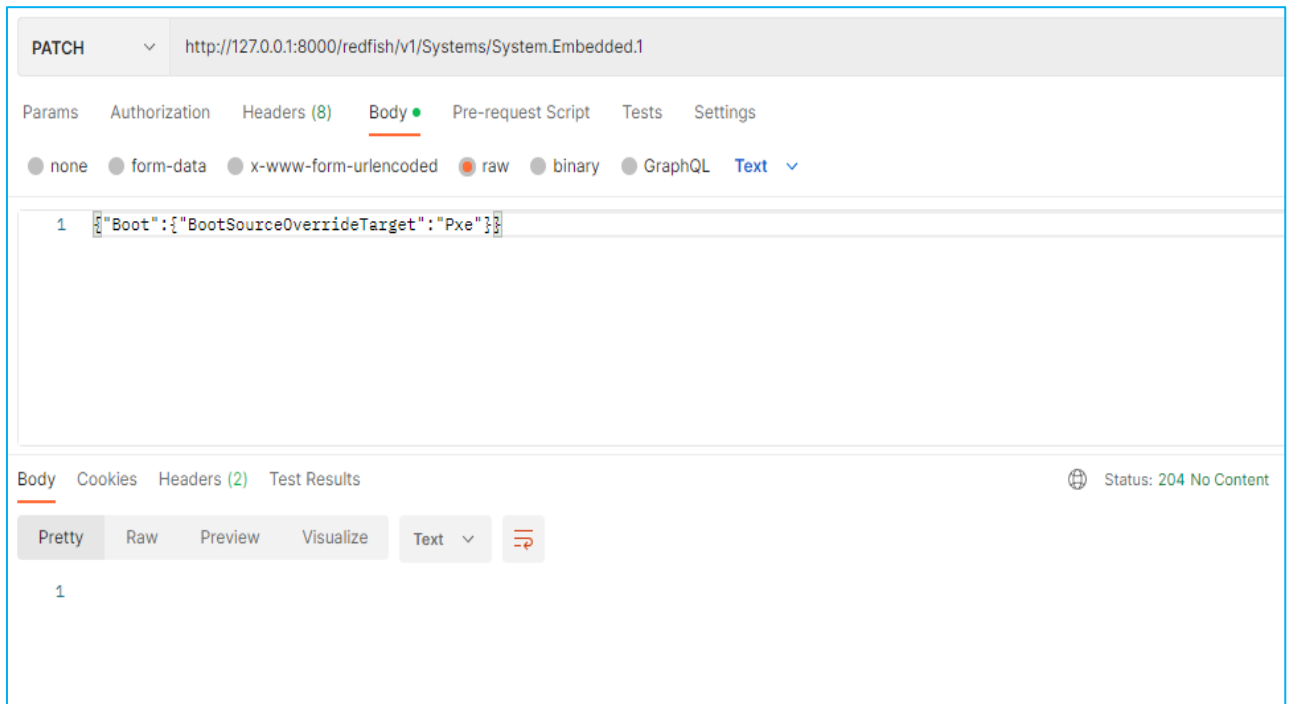


Figure 4 PATCH request for BootSourceOverrideTarget

```
PATCH: Data: {'Boot': {'BootSourceOverrideTarget': 'Pxe'}}
text/plain
{'Boot': {'BootSourceOverrideTarget': 'Pxe'}}
{'@Redfish.Settings': {'@odata.context':
'/redfish/v1/$metadata#Settings.Settings', '@odata.type':
'#Settings.v1_3_5.Settings', 'SettingsObject': {'@odata.id':
'/redfish/v1/Systems/System.Embedded.1/Settings'}, 'SupportedApplyTimes':
['OnReset']}, '@odata.context':
'/redfish/v1/$metadata#ComputerSystem.ComputerSystem', '@odata.id':
'/redfish/v1/Systems/System.Embedded.1', '@odata.type':
'#ComputerSystem.v1_20_1.ComputerSystem', 'Actions': {'#ComputerSystem.Reset':
{'target': '/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.Reset',
'ResetType@Redfish.AllowableValues': ['On', 'ForceOff', 'ForceRestart',
'GracefulRestart', 'GracefulShutdown', 'PushPowerButton', 'Nmi',
'PowerCycle']}}, 'AssetTag': '123456', 'Bios': {'@odata.id':
'/redfish/v1/Systems/System.Embedded.1/Bios'}, 'BiosVersion': '1.12.1',
'BootProgress': {'LastState': 'OSRunning'}, 'Boot': {'BootOptions':
{'@odata.id': '/redfish/v1/Systems/System.Embedded.1/BootOptions'},
```

```

'Certificates': {'@odata.id':
'/redfish/v1/Systems/System.Embedded.1/Boot/Certificates'}, 'BootOrder':
['Boot0010', 'Boot000E', 'Boot000D', 'Boot0001', 'Boot0003'],
'BootOrder@odata.count': 5, 'BootSourceOverrideEnabled': 'Disabled',
'BootSourceOverrideMode': 'UEFI', 'BootSourceOverrideTarget': 'Pxe',
'UefiTargetBootSourceOverride': '3A191845-5F86-4E78-8FCE-C4CFF59F9DAA',
'BootSourceOverrideTarget@Redfish.AllowableValues': ['None', 'Pxe', 'Floppy',
'Cd', 'Hdd', 'BiosSetup', 'Utilities', 'UefiTarget', 'SDCard', 'UefiHttp'],
'StopBootOnFault': 'Never'}, 'Description': 'Computer System which represents a
machine (physical or virtual) and the local resources such as memory, cpu and
other devices that can be accessed from that machine.', 'EthernetInterfaces':
{'@odata.id': '/redfish/v1/Systems/System.Embedded.1/EthernetInterfaces'},
'GraphicalConsole': {'ConnectTypesSupported': ['KVMIP'],
'ConnectTypesSupported@odata.count': 1, 'MaxConcurrentSessions': 6,
'ServiceEnabled': True}, 'HostName': 'localhost.ept.adc.delllabs.net',
'HostWatchdogTimer': {'FunctionEnabled': False, 'Status': {'State': 'Disabled'}},
'TimeoutAction': 'None'}, 'HostingRoles': [], 'HostingRoles@odata.count': 0,
'Id': 'System.Embedded.1', 'IndicatorLED': 'Lit',
'IndicatorLED@Redfish.Deprecated': 'Please migrate to use
LocationIndicatorActive property', 'Links': {'Chassis': [{'@odata.id':
'/redfish/v1/Chassis/System.Embedded.1'}]}, 'Chassis@odata.count': 1, 'CooledBy':
[{'@odata.id': '/redfish/v1/Chassis/System.Embedded.1/Thermal#/Fans/0'}]}
127.0.0.1 - - [01/Feb/2024 10:11:17] "PATCH
/redfish/v1/Systems/System.Embedded.1 HTTP/1.1" 204 -

```

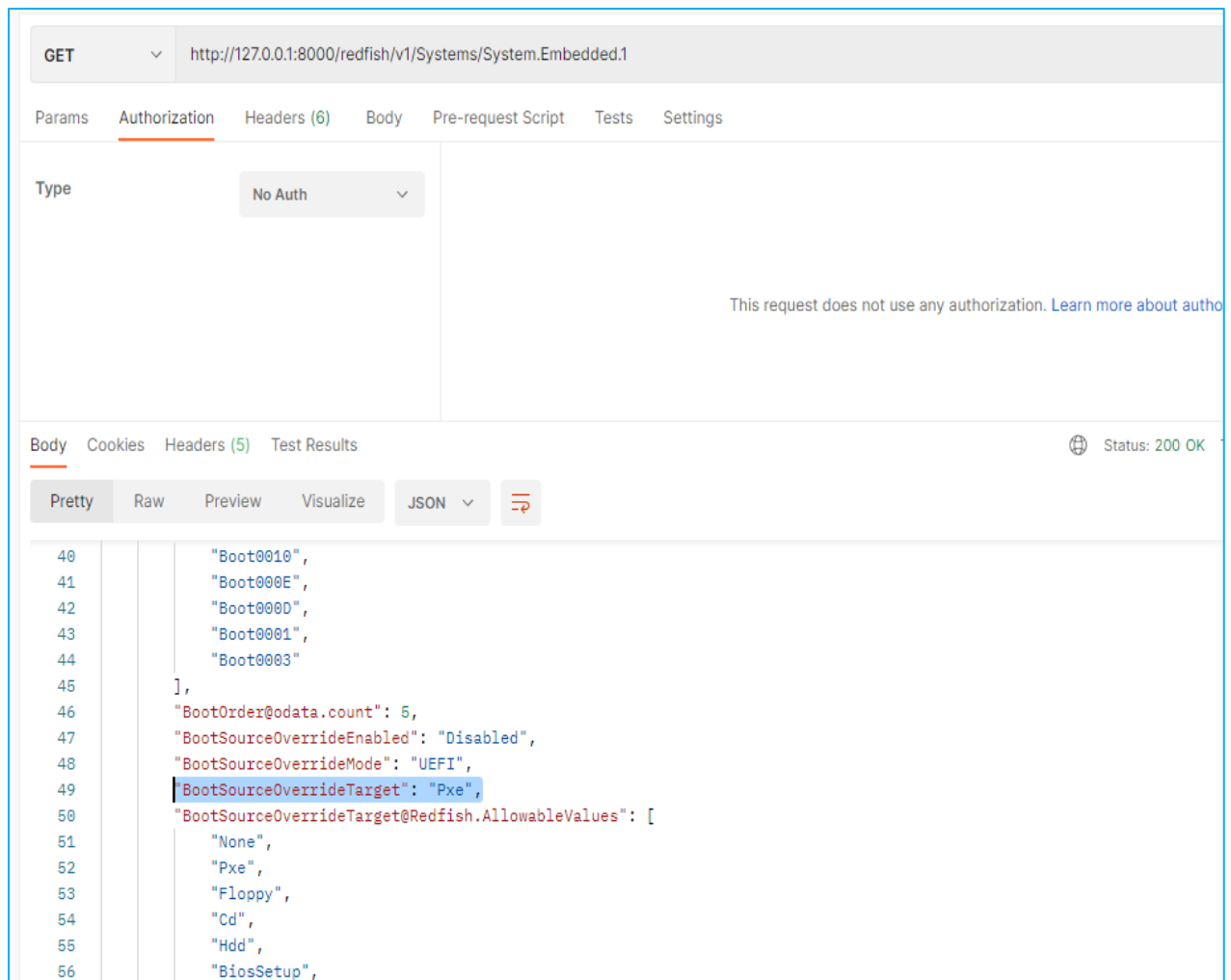


Figure 5 GET request to see the updated property value

```

127.0.0.1 - - [06/Feb/2024 12:55:06] "GET
/redfish/v1/UpdateService/FirmwareInventory?$expand=*$levels=1) HTTP/1.1" 200 -
('GET', '/redfish/v1/Systems/System.Embedded.1')
GET: Headers: User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: b9674463-ad22-47e8-9743-7901f355e0a7
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

```

2.2.4 PATCH Request for OEM system attribute

The following example indicates the GET operation for current iDRAC system `PowerCapSettings`, change values using the PATCH operation, and confirm new values along with mockup server logs.

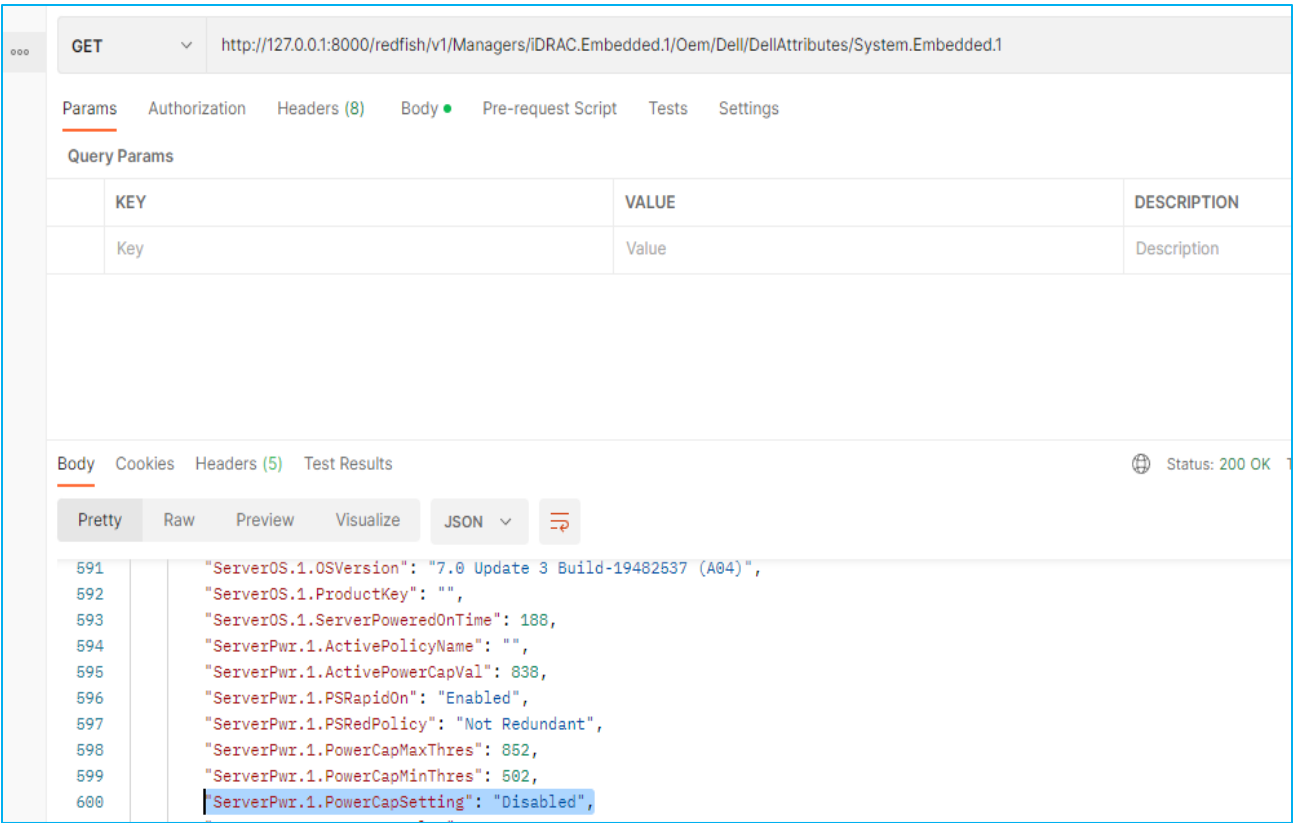


Figure 6 GET operation for PowerCapSettings

```

('GET',
'/redfish/v1/Managers/iDRAC.Embedded.1/Oem/Dell/DellAttributes/System.Embedded.1
')

GET: Headers: Content-Type: text/plain
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 12c4eb52-6e37-4a2f-85a1-c6ddcd0e3ed5
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 53

```

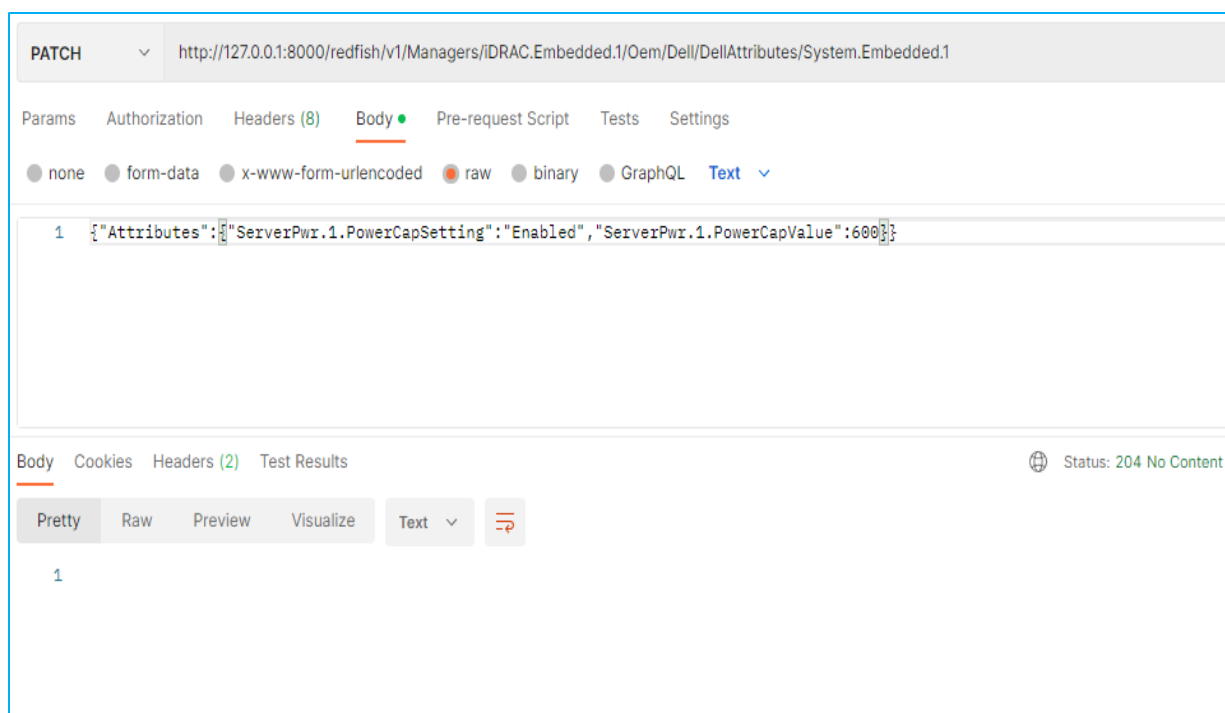


Figure 7 PATCH operation to change value

```
127.0.0.1 - - [06/Feb/2024 13:34:53] "GET
/redfish/v1/Managers/iDRAC.Embedded.1/Oem/Dell/DellAttributes/System.Embedded.1
HTTP/1.1" 200 -
```

```
PATCH: Headers: Content-Type: text/plain
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: c3843b5d-78a0-4ee9-aa60-10448c8a3a16
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 88
```

```
PATCH: Data: {'Attributes': {'ServerPwr.1.PowerCapSetting': 'Enabled',
'ServerPwr.1.PowerCapValue': 600}}
text/plain
{'Attributes': {'ServerPwr.1.PowerCapSetting': 'Enabled',
'ServerPwr.1.PowerCapValue': 600}}
```

GET `http://127.0.0.1:8000/redfish/v1/Managers/iDRAC.Embedded.1/Oem/Dell/DellAttributes/System.Embedded.1`

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results ⌐ Status: 200 OK

Pretty Raw Preview Visualize JSON ≡

```

587   "ServerOS.1.HostName": "localhost.ept.adc.delllabs.net",
588   "ServerOS.1.InstallCompletedTime": "",
589   "ServerOS.1.OEMOSVersion": "",
590   "ServerOS.1.OSName": "DellEMC-VMware ESXi",
591   "ServerOS.1.OSVersion": "7.0 Update 3 Build-19482537 (A04)",
592   "ServerOS.1.ProductKey": "",
593   "ServerOS.1.ServerPoweredOnTime": 188,
594   "ServerPwr.1.ActivePolicyName": "",
595   "ServerPwr.1.ActivePowerCapVal": 838,
596   "ServerPwr.1.PSRapidOn": "Enabled",
597   "ServerPwr.1.PSRedPolicy": "Not Redundant",
598   "ServerPwr.1.PowerCapMaxThres": 852,
599   "ServerPwr.1.PowerCapMinThres": 502,
600   "ServerPwr.1.PowerCapSetting": "Enabled",
601   "ServerPwr.1.PowerCapValue": 600,
602   "ServerPwr.1.Poweredbyparent": "SelfPowered",
603   "ServerPwr.1.RapidOnPrimaryPSU": "PSU1",
604   "ServerPwrMon.1.AccumulativePower": 2575385,

```

Figure 8 Confirm mockup server logs

```

('GET',
'/redfish/v1/Managers/iDRAC.Embedded.1/Oem/Dell/DellAttributes/System.Embedded.1')

GET: Headers: Content-Type: text/plain
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: c124e851-b34e-4b43-b9e4-db46fe812f8d
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 53

```

2.2.5 PATCH request to create iDRAC user

The following example shows the request to create a new iDRAC user (id 4). The `GET` operation is leveraged to validate the properties of a newly created and enabled user:

GET <http://127.0.0.1:8000/redfish/v1/AccountService/Accounts/4>

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON ☒

```

1  {
2    "@odata.context": "/redfish/v1/$metadata#ManagerAccount.ManagerAccount",
3    "@odata.etag": "\"917975\"",
4    "@odata.id": "/redfish/v1/AccountService/Accounts/4",
5    "@odata.type": "#ManagerAccount.v1_10_0.ManagerAccount",
6    "AccountTypes": [
7      "Redfish",
8      "SNMP",
9      "OEM",
10     "HostConsole",
11     "ManagerConsole",
12     "IPMI",
13     "KVMIP",
14     "VirtualMedia",
15     "WebUI"
16   ],
17   "Description": "User Account",
18   "Enabled": false,
19   "Id": "4",
20   "Keys": {
21     "@odata.id": "/redfish/v1/AccountService/Accounts/4/Keys"
22   },
23   "Links": {
24     "Role": {
25       "@odata.id": "/redfish/v1/AccountService/Roles/None"
26     }
27   }
28 }

```

Figure 9 The GET operation to create new user

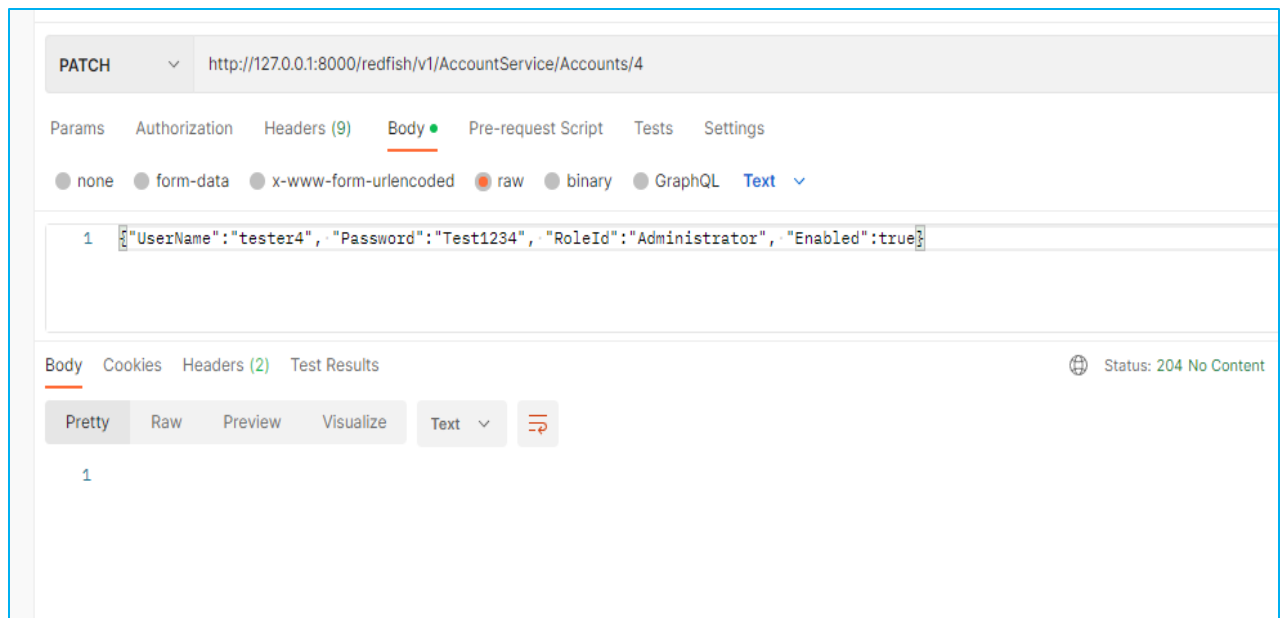


Figure 10 The PATCH operation to validate new user

GET

http://127.0.0.1:8000/redfish/v1/AccountService/Accounts/4

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

"@odata.context": "/redfish/v1/\$metadata#ManagerAccount.ManagerAccount",

"@odata.etag": "\"917975\"",

"@odata.id": "/redfish/v1/AccountService/Accounts/4",

"@odata.type": "#ManagerAccount.v1_10_0.ManagerAccount",

"AccountTypes": [

"Redfish",

"SNMP",

"OEM",

"HostConsole",

"ManagerConsole",

"IPMI",

"KVMIP",

"VirtualMedia",

"WebUI"

],

"Description": "User Account",

"Enabled": true,

"Id": "4",

"Keys": {

"@odata.id": "/redfish/v1/AccountService/Accounts/4/Keys"

Figure 11 The GET operation to enable new user

```

37      "Oem": {
38        "Dell": {
39          "@odata.type": "#DellManagerAccount.v1_0_0.DellManagerAccount",
40          "SNMPv3PassphraseEnabled": "Disabled"
41        }
42      },
43      "Password": "Test1234",
44      "PasswordChangeRequired": false,
45      "PasswordExpiration": null,
46      "RoleId": "Administrator",
47      "SNMP": {
48        "AuthenticationKey": null,
49        "AuthenticationKeySet": false,
50        "AuthenticationProtocol": "HMAC_SHA96",
51        "EncryptionKey": null,
52        "EncryptionKeySet": false,
53        "EncryptionProtocol": "CFB128_AES128"
54      },
55      "StrictAccountTypes": false,
56      "UserName": "tester4"
57    }

```

Figure 12 The GET operation to validate new user

Note—The `Password` property is displayed only for Redfish mockup server. An actual iDRAC does not display passwords in clear text.

2.2.6 PATCH request to change iDRAC user password

The following example indicates to modify iDRAC user (id 4) password using a `PATCH` operation:

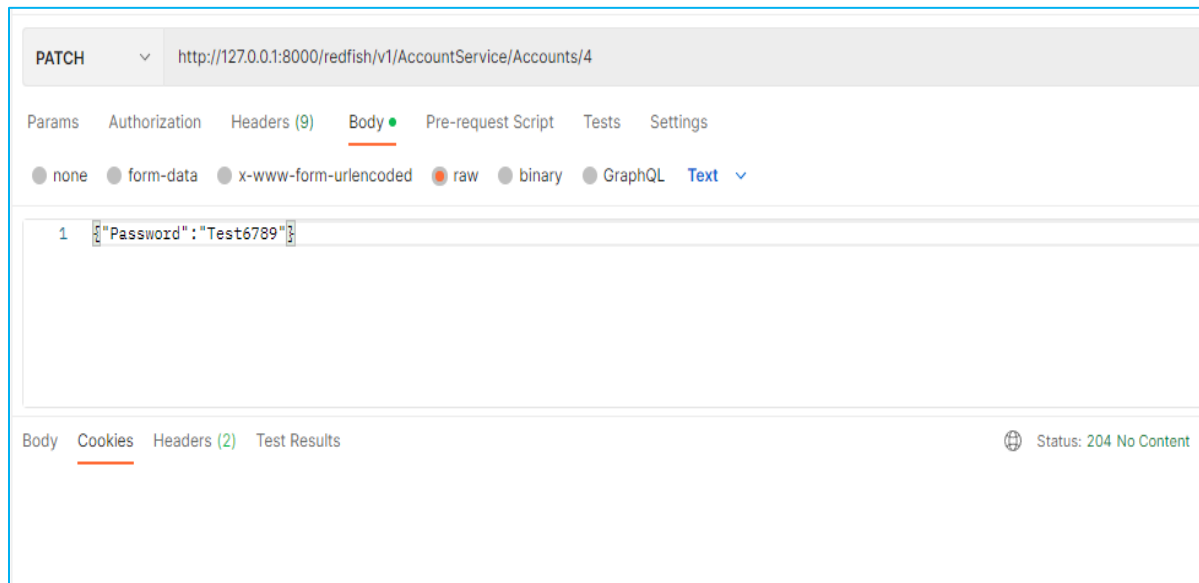


Figure 13 The PATCH operation to change password

2.2.7 POST request for Distributed Management Task Force (DMTF) action InsertMedia

The following example indicates the `POST` request using action `InsertMedia` to attach virtual media ISO image file:

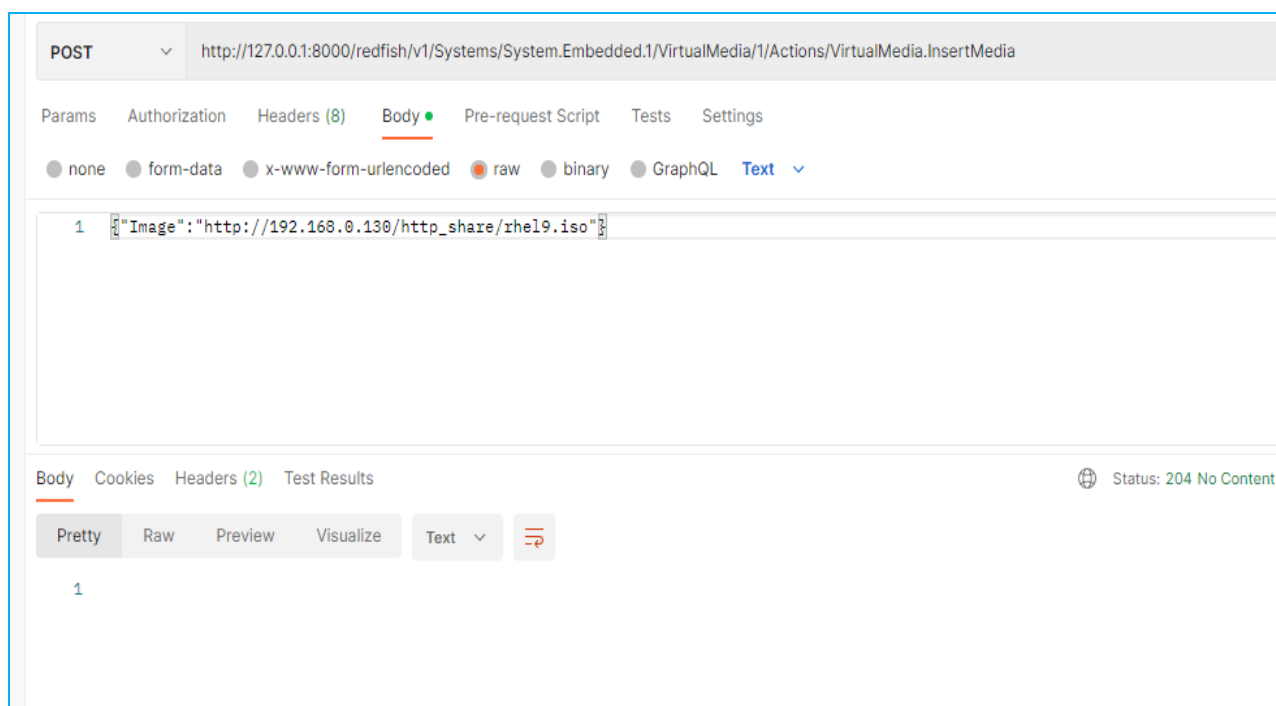


Figure 14 The POST request for DMTF action

```
POST: Data: {}
127.0.0.1 - - [01/Feb/2024 10:27:18] "POST
/redfish/v1/Systems/System.Embedded.1/VirtualMedia/1/Actions/VirtualMedia.Insert
Media HTTP/1.1" 204 -
POST: Headers: Content-Type: text/plain
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 16dd3465-3d42-4ddb-afee-a95da1c30bac
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 53
```

```
POST: Data: {'Image': 'http://192.168.0.130/http_share/rhel9.iso'}
127.0.0.1 - - [01/Feb/2024 10:28:06] "POST
/redfish/v1/Systems/System.Embedded.1/VirtualMedia/1/Actions/VirtualMedia.Insert
Media HTTP/1.1" 204 -
```

2.2.8 POST Request for OEM action ExportSystemConfiguration

The following example indicates the running OEM action `ExportSystemConfiguration` to export iDRAC system configuration data locally along with mockup server log data:

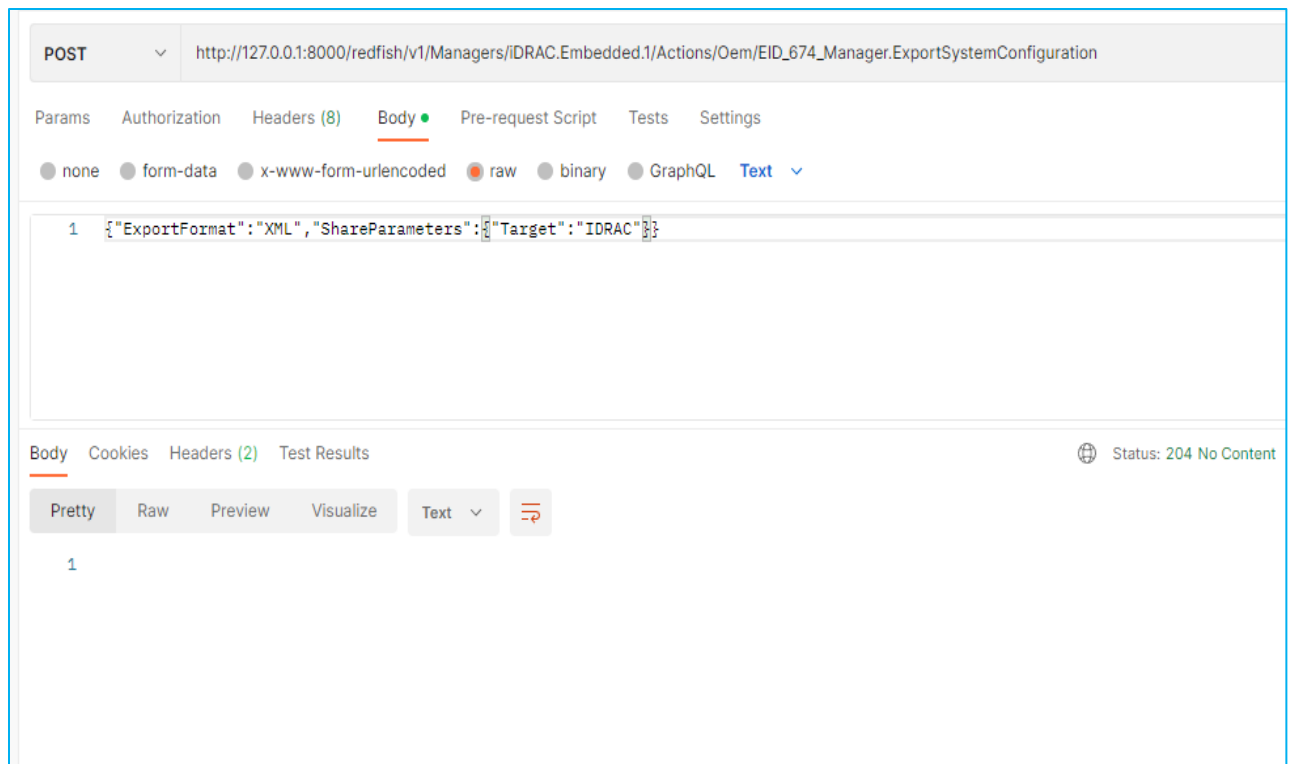


Figure 15 The POST operation to export iDRAC

```
127.0.0.1 - - [06/Feb/2024 13:26:27] "POST
/redfish/v1/Managers/iDRAC.Embedded.1/Actions/Oem/EID_674_Manager.ExportSystemCo
nfiguration HTTP/1.1" 204 -
    POST: Headers: Content-Type: text/plain
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 13ef4539-f8b5-4a2e-88d1-9c2742f57bdb
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 59
```

```
    POST: Data: {'ExportFormat': 'XML', 'ShareParameters': {'Target': 'iDRAC'}}
127.0.0.1 - - [06/Feb/2024 13:27:11] "POST
/redfish/v1/Managers/iDRAC.Embedded.1/Actions/Oem/EID_674_Manager.ExportSystemCo
nfiguration HTTP/1.1" 204 -
```

2.2.9 POST request to create new session

The following example indicates the GET operation for current active sessions, create new Redfish session, and the GET session details along with mockup server logs:

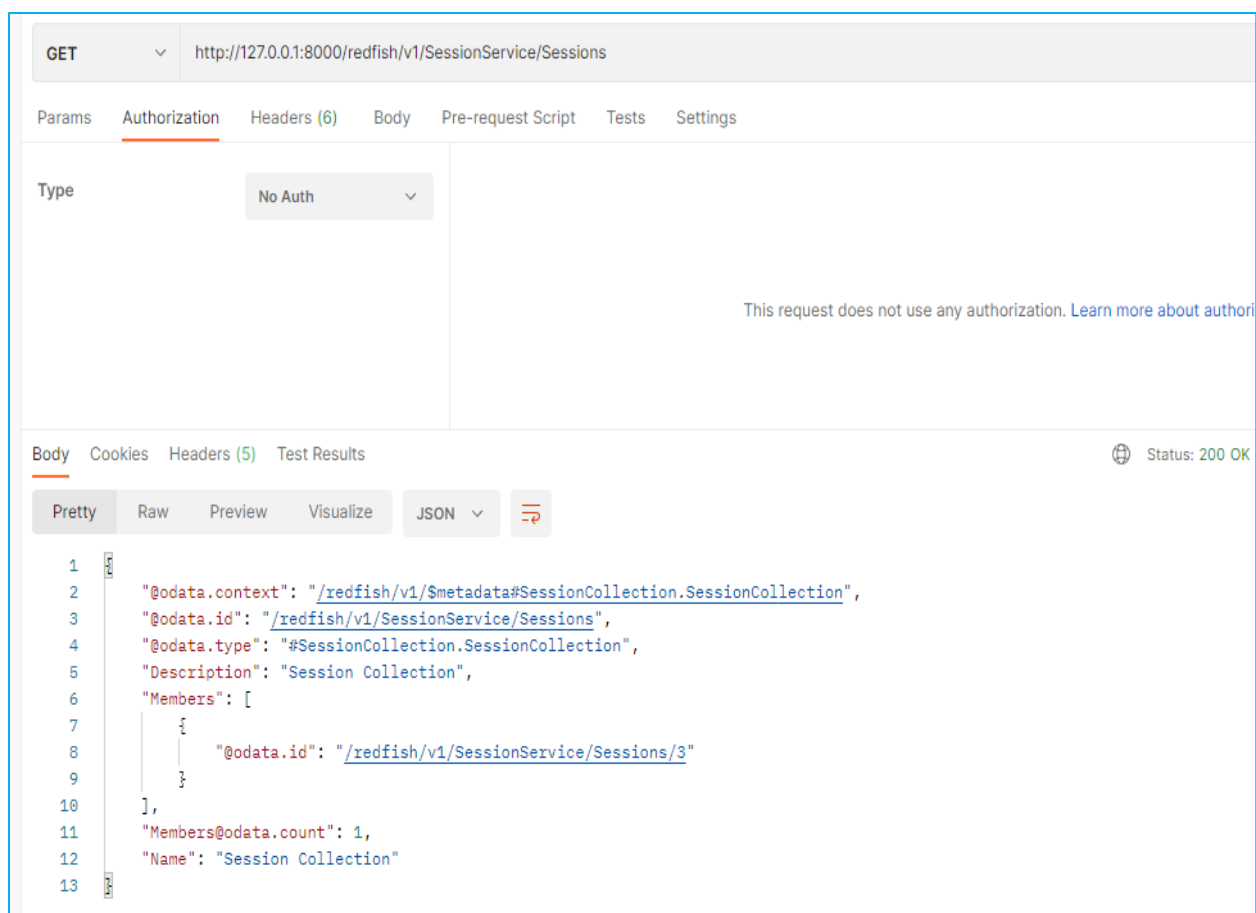


Figure 16 The GET operation to view active sessions

```

('GET', '/redfish/v1/SessionService/Sessions')
GET: Headers: User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: f581274f-f930-491a-b758-bdb5360716ec
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

```

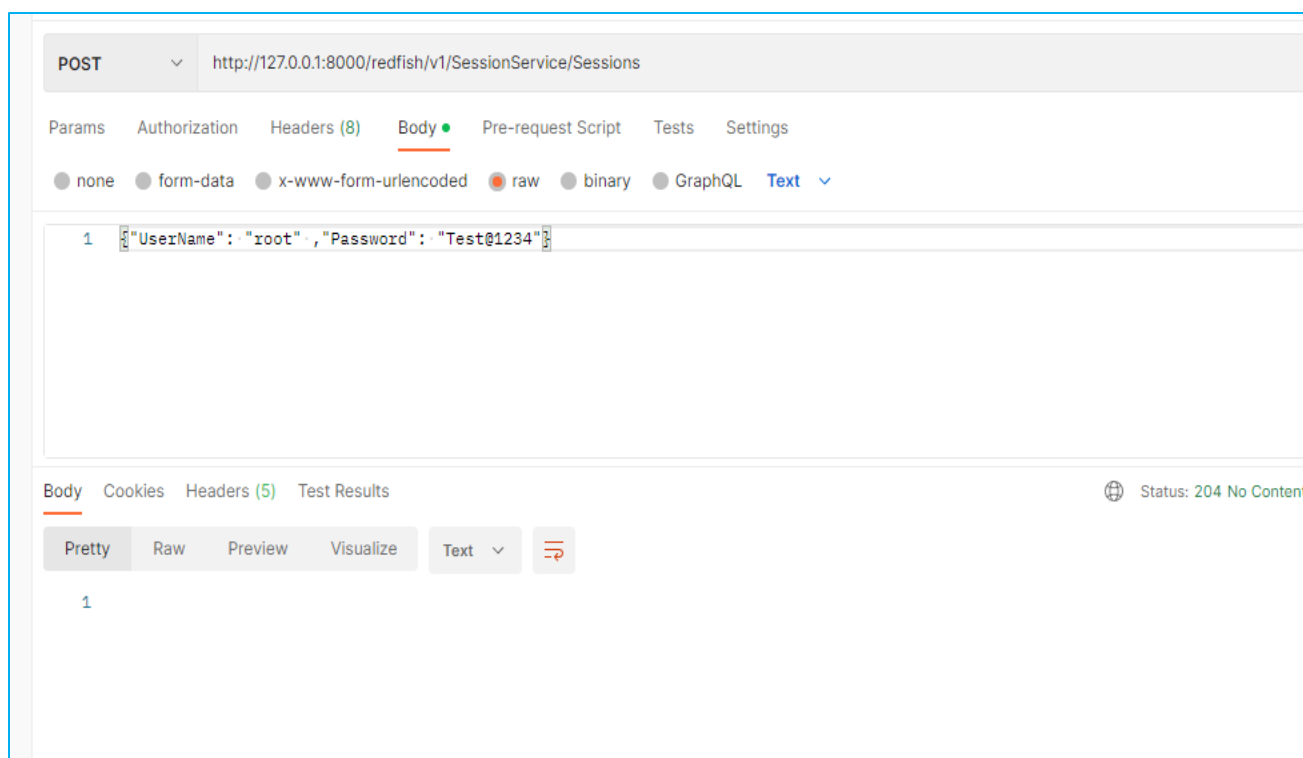


Figure 17 The POST request for active sessions

```
127.0.0.1 - - [06/Feb/2024 14:11:45] "GET /redfish/v1/SessionService/Sessions
HTTP/1.1" 200 -
```

```
POST: Headers: Content-Type: text/plain
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 64806302-13e0-499a-819b-603e03edfaf2
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 45
```

```
POST: Data: {'UserName': 'root', 'Password': 'Test@1234'}
{'UserName': 'root', 'Password': 'Test@1234'}
<class 'dict'>
C:\Python310\iDRAC_mockup_client/redfish/v1/SessionService/Sessions/1/index.json
127.0.0.1 - - [06/Feb/2024 14:16:01] "POST /redfish/v1/SessionService/Sessions
HTTP/1.1" 204 -
```

The screenshot shows a web browser interface for a REST client. The URL bar displays `http://127.0.0.1:8000/redfish/v1/SessionService/Sessions`. The 'Authorization' tab is selected, showing 'No Auth'. The 'Body' tab is also visible, displaying a JSON response. The status bar at the bottom right indicates 'Status: 200 OK'.

```

1  {
2    "@odata.context": "/redfish/v1/$metadata#SessionCollection.SessionCollection",
3    "@odata.id": "/redfish/v1/SessionService/Sessions",
4    "@odata.type": "#SessionCollection.SessionCollection",
5    "Description": "Session Collection",
6    "Members": [
7      {
8        "@odata.id": "/redfish/v1/SessionService/Sessions/3"
9      },
10     {
11       "@odata.id": "/redfish/v1/SessionService/Sessions/1"
12     }
13   ],
14   "Members@odata.count": 2,
15   "Name": "Session Collection"
16 }

```

Figure 18 The GET session information

```

127.0.0.1 - - [06/Feb/2024 14:16:44] "GET /redfish/v1/SessionService/Sessions
HTTP/1.1" 200 -
('GET', '/redfish/v1/SessionService/Sessions')
GET: Headers: User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: b273a0ca-d407-499b-aeb8-435f0fc10660
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

```

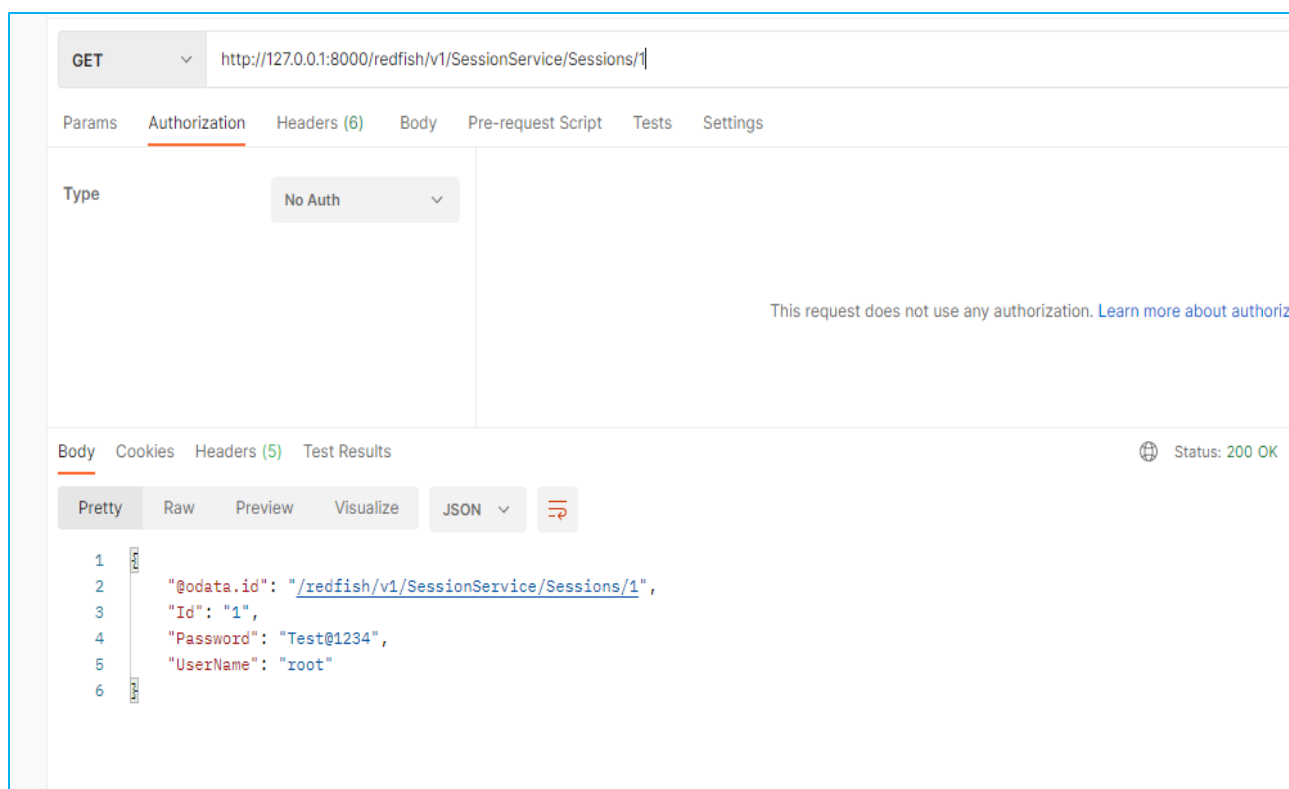



Figure 19 GET command to view mockup server logs

```

127.0.0.1 - - [06/Feb/2024 14:16:44] "GET /redfish/v1/SessionService/Sessions
HTTP/1.1" 200 -
('GET', '/redfish/v1/SessionService/Sessions/1')
GET: Headers: User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: b273a0ca-d407-499b-aeb8-435f0fc10660
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

```

2.2.10 DELETE request to delete DMTF virtual disk

The following example indicates the GET command to view the current virtual disks for storage controller, delete the virtual disk, and ensure that virtual disk no longer exists along with mockup server logs.

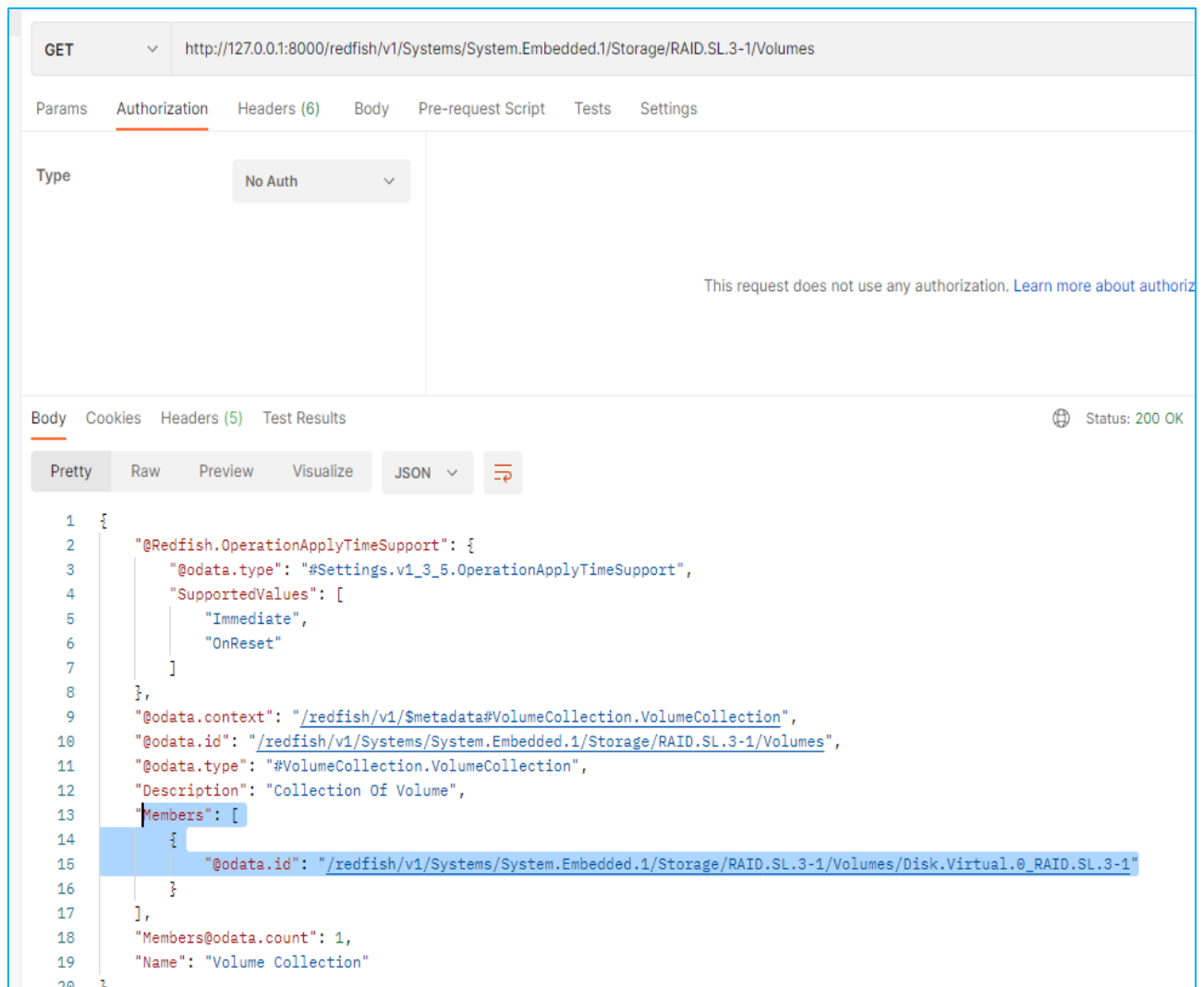


Figure 20 The GET operation to see current virtual disk

```

127.0.0.1 - - [06/Feb/2024 13:48:58] "GET
/redfish/v1/Systems/System.Embedded.1/Storage/RAID.SL.3-
1/Volumes/Disk.Virtual.0_RAID.SL.3-1 HTTP/1.1" 200 -
('GET', '/redfish/v1/Systems/System.Embedded.1/Storage/RAID.SL.3-1/Volumes')
GET: Headers: Content-Type: text/plain
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: a0eecfe4-c7ca-4b7a-a5fb-9dd9343815f1
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 57

```

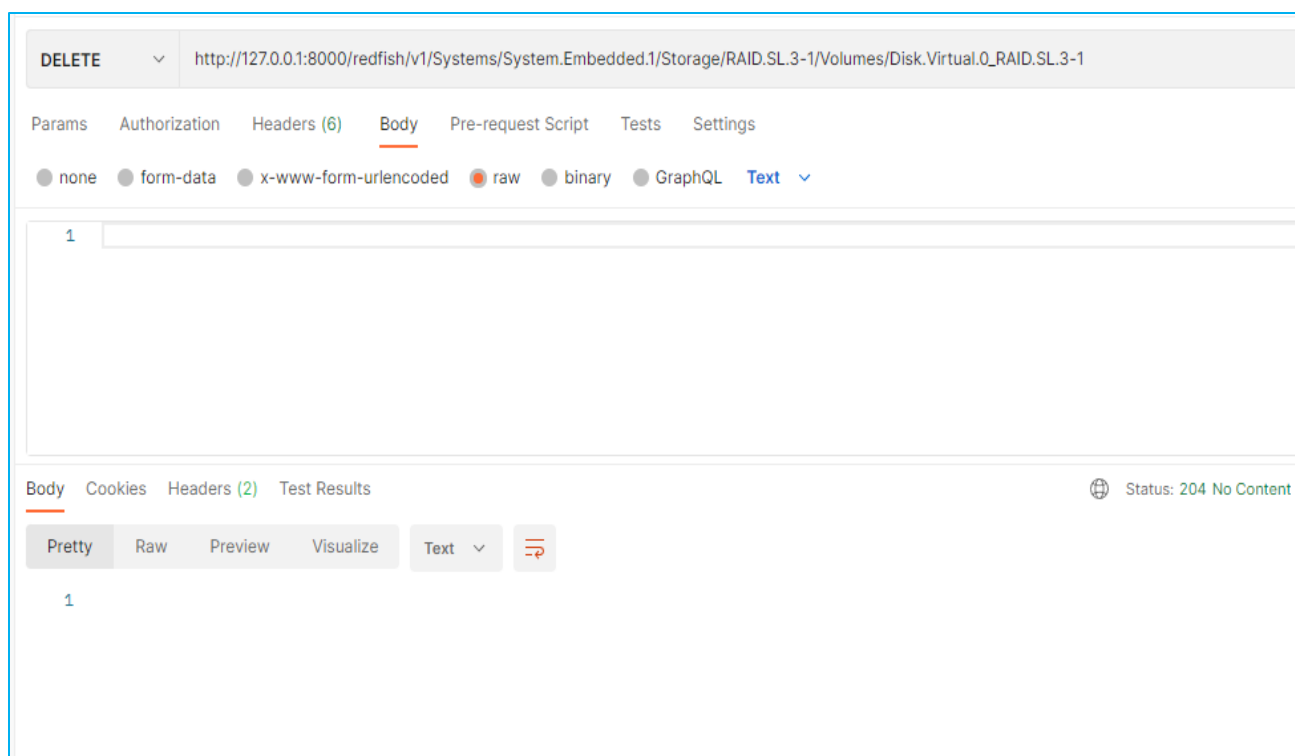


Figure 21 The DELETE operation

```

127.0.0.1 - - [06/Feb/2024 13:49:12] "GET
/redfish/v1/Systems/System.Embedded.1/Storage/RAID.SL.3-1/Volumes HTTP/1.1" 200
-
DELETE: Headers: User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 155f8cd0-7f91-415d-b18a-db9a29cccc82
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

127.0.0.1 - - [06/Feb/2024 13:51:13] "DELETE
/redfish/v1/Systems/System.Embedded.1/Storage/RAID.SL.3-
1/Volumes/Disk.Virtual.0_RAID.SL.3-1 HTTP/1.1" 204 -

```

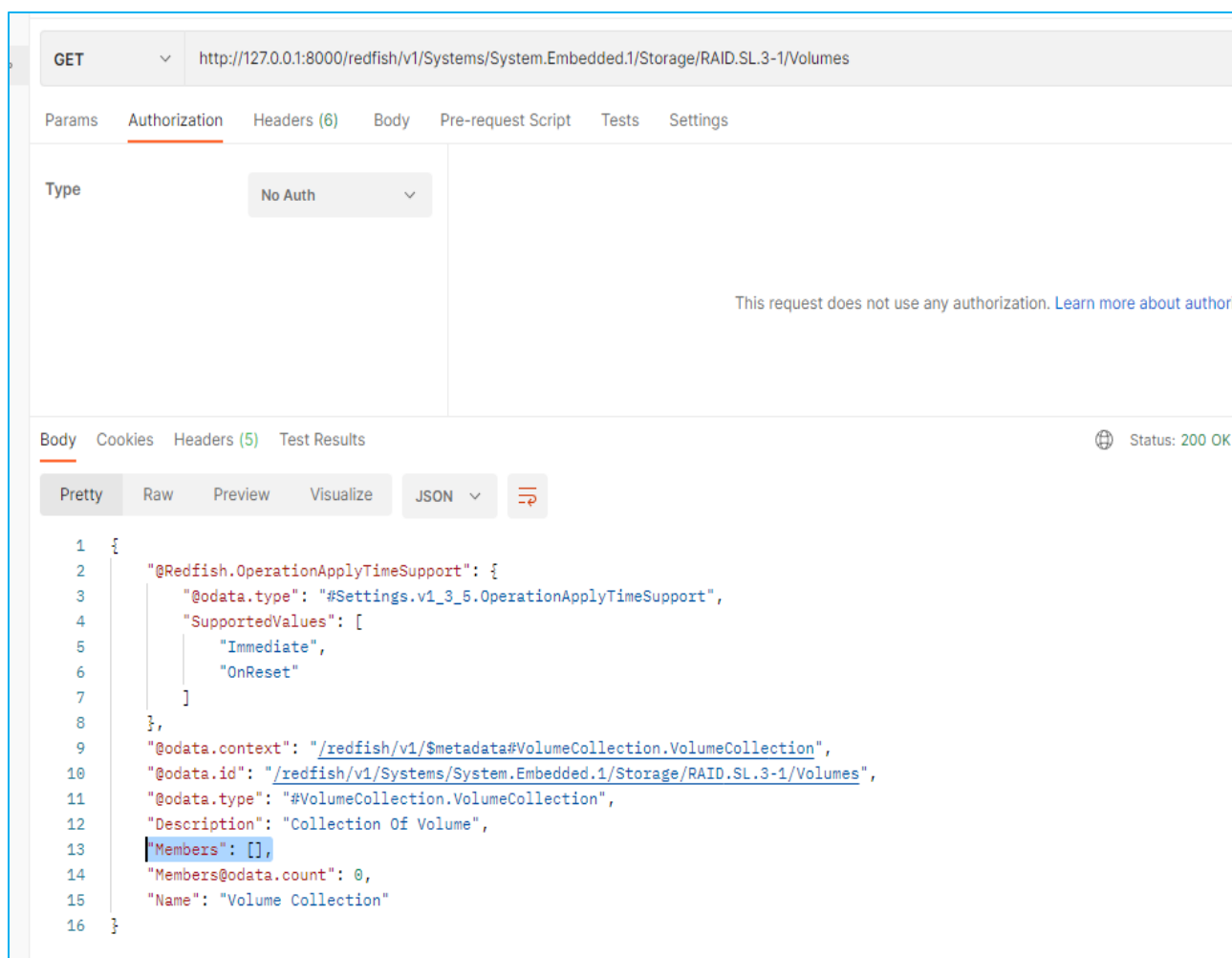


Figure 22 The GET command for mockup server logs

```

127.0.0.1 - - [06/Feb/2024 13:51:13] "DELETE
/redfish/v1/Systems/System.Embedded.1/Storage/RAID.SL.3-
1/Volumes/Disk.Virtual.0_RAID.SL.3-1 HTTP/1.1" 204 -
('GET', '/redfish/v1/Systems/System.Embedded.1/Storage/RAID.SL.3-1/Volumes')
GET: Headers: User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 8fc87501-0946-4279-aa53-874ebc638116
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

```

3 Simulate streaming iDRAC Redfish events workflow

The Redfish mockup server along with Redfish Event Listener enables you to simulate streaming iDRAC alert events. This section explains the following tasks:

- [Configure Redfish Event Listener](#)
- [Create subscription](#)
- [Get subscription details](#)
- [Start Redfish Event Listener](#)
- [Submit test event](#)
- [Confirm Redfish event listener receives the event](#)

For more information about the workflow, see the GitHub Redfish Event Listener:

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

3.1 Configure Redfish Event Listener

Edit the Redfish event listener `config.ini` and ensure the following properties are defined:

- `ListenerPort = 80`
- `UseSSL = off`
- `Destination = http://{client IP that will be running Redfish Event Listener}:80`
- `ServerIPs = http://127.0.0.1:8000`
- `certcheck = off`

Following is the example of an edited `config.ini`:

```
[Information]
Updated = April 24, 2017
Description = Redfish Event Listener Tool
[SystemInformation]
ListenerIP = 0.0.0.0
ListenerPort = 80
UseSSL = off
[CertificateDetails]
certfile = cert.pem
keyfile = server.key
[SubscriptionDetails]
Destination = http://192.168.0.130:80
Context = Public
Protocol = Redfish
SubscriptionURI = /redfish/v1/EventService/Subscriptions
[ServerInformation]
ServerIPs = http://127.0.0.1:8000
UserNames = root
Passwords = calvin
certcheck = off
```

3.2 Create a subscription

Create a subscription using same destination URI string you provided in the `config.ini` file.

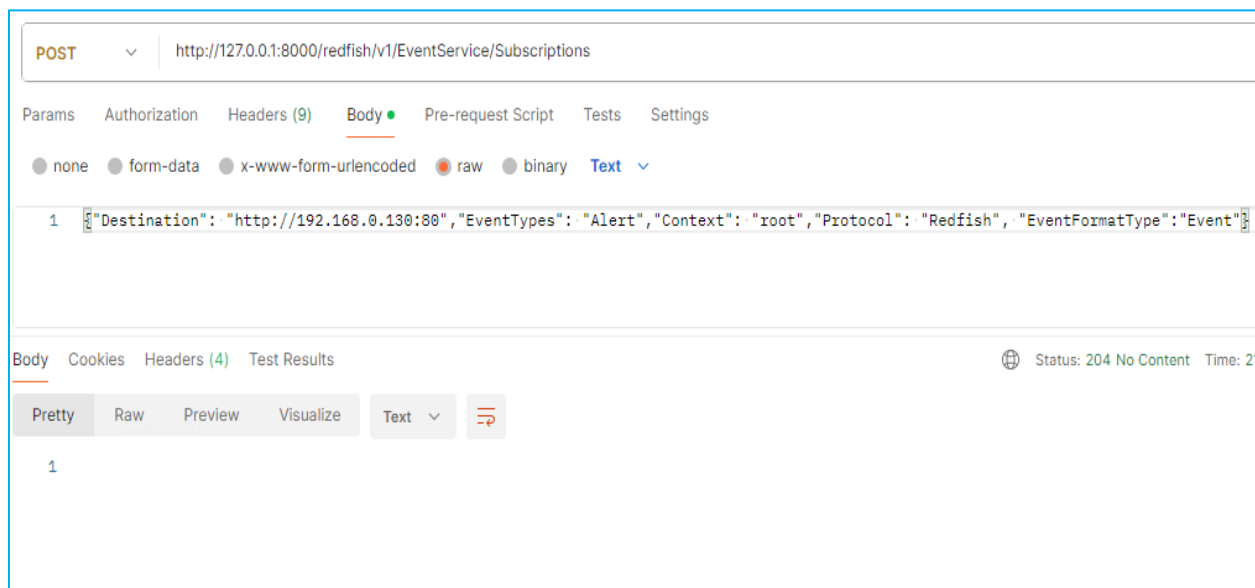


Figure 23 Create a subscription

3.3 Get subscription information

Get subscription information to validate the new subscriptions that are created.

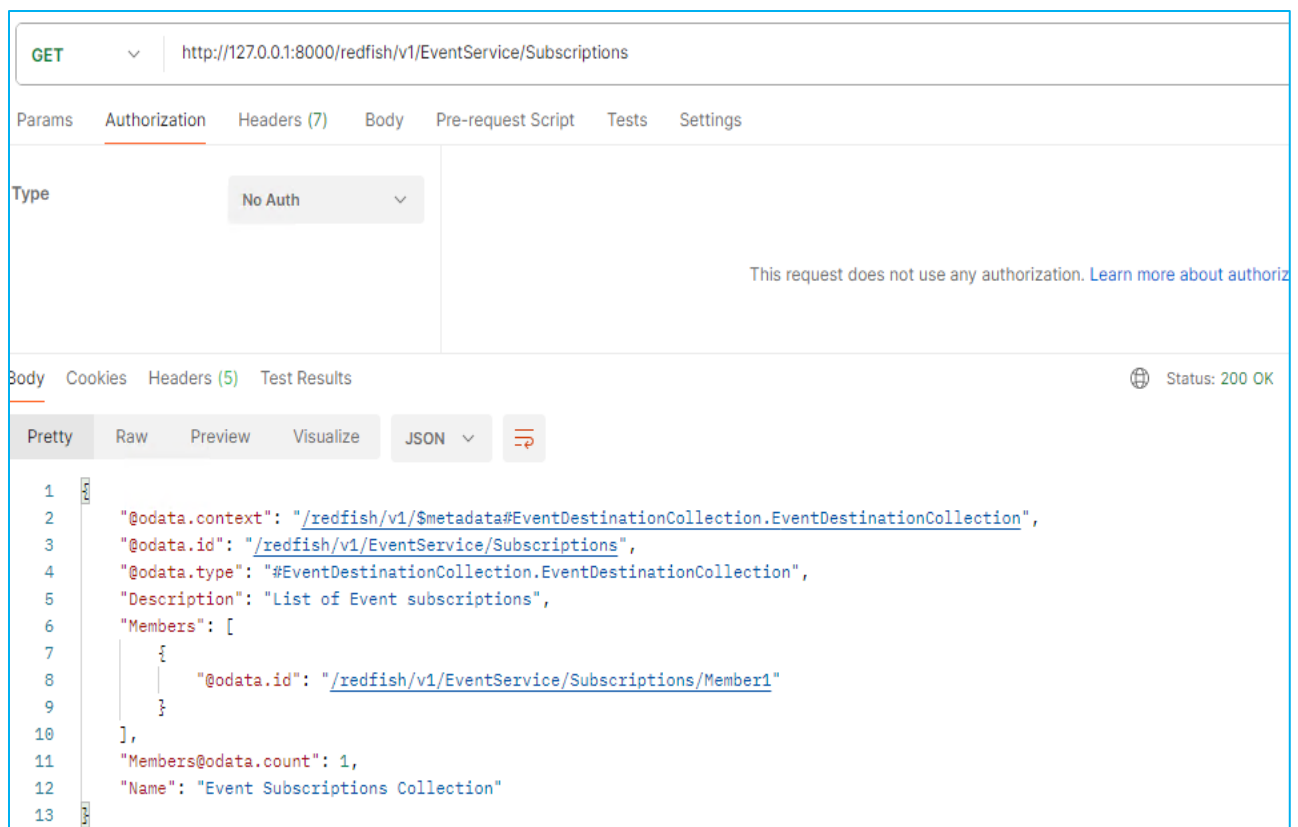


Figure 24 Subscription information

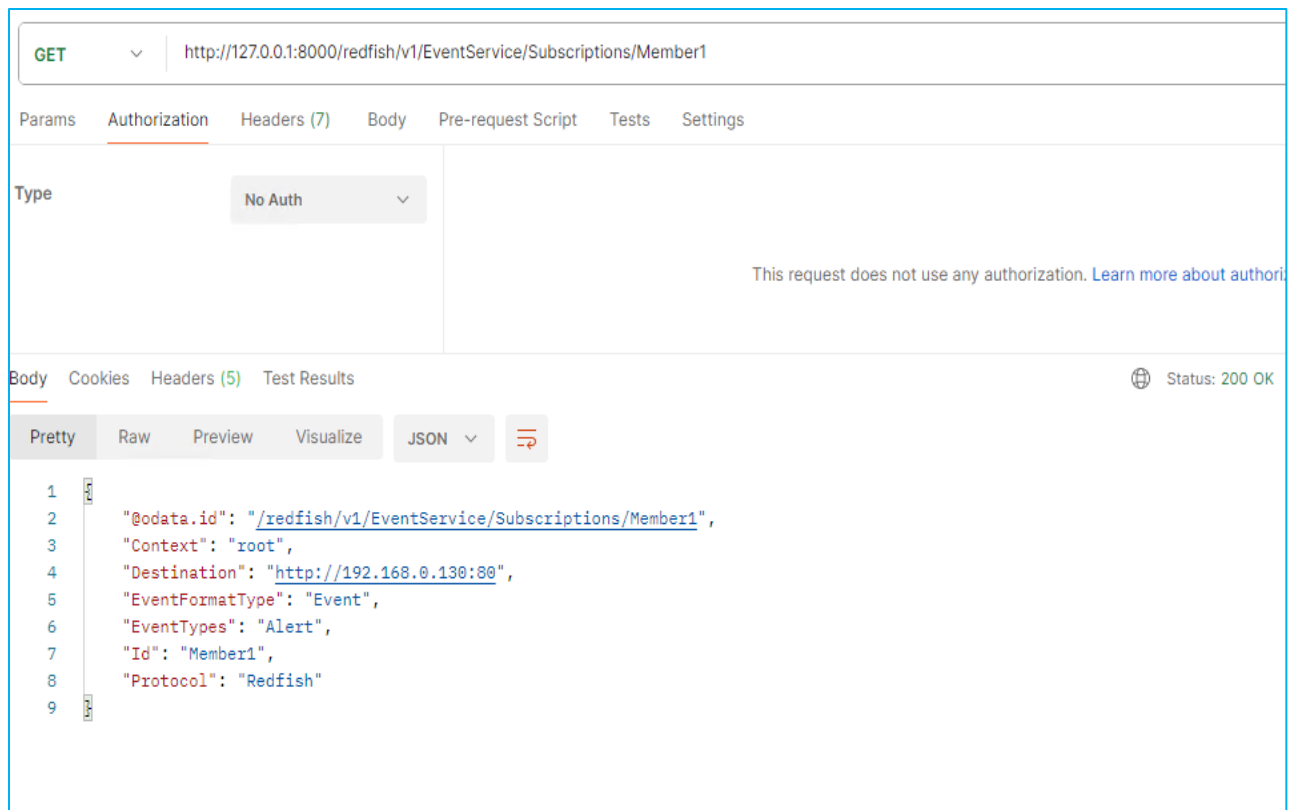


Figure 25 Validate the subscription

3.4 Start the Redfish Event Listener

Start the Redfish Event Listener. If you see any errors, see the GitHub readme page.

```

C:\Python310>python RedfishEventListener_v1.py
Redfish Event Listener v1.1.3
ServerIP:: http://127.0.0.1:8000
UserName:: root
Attempt 1 of /redfish/v1/
Response Time for GET to /redfish/v1/: 0.021674100076779723 seconds.
Attempt 1 of /redfish/v1/SessionService/Sessions
Response Time for POST to /redfish/v1/SessionService/Sessions:
0.009719900088384748 seconds.
Login returned code 204:
Attempt 1 of /redfish/v1/
Response Time for GET to /redfish/v1/: 0.010551500134170055 seconds.
Attempt 1 of /redfish/v1/EventService
Response Time for GET to /redfish/v1/EventService: 0.011112900217995048 seconds.
Attempt 1 of /redfish/v1/EventService/Subscriptions
Response Time for POST to /redfish/v1/EventService/Subscriptions:
0.009794200072064996 seconds.
Subscription is successful for http://127.0.0.1:8000,
/redfish/v1/EventService/Subscriptions/Member2
Service responded with invalid JSON at URI
/redfish/v1/EventService/Subscriptions

```



```
Continuing with Listener.
Listening on 0.0.0.0:80 via HTTP
Press Ctrl-C to close program
..
```

3.5 Submit a test event

Submit a test event using any valid iDRAC message ID from the message registry. To get supported message IDs, run the GET operation on `redfish/v1/Registries/Messages/EEMIRegistry`.

Note—In the following example, all body properties listed are required to submit a test event. Else, 400 Error Code is displayed. For the property values, enter any string for testing.

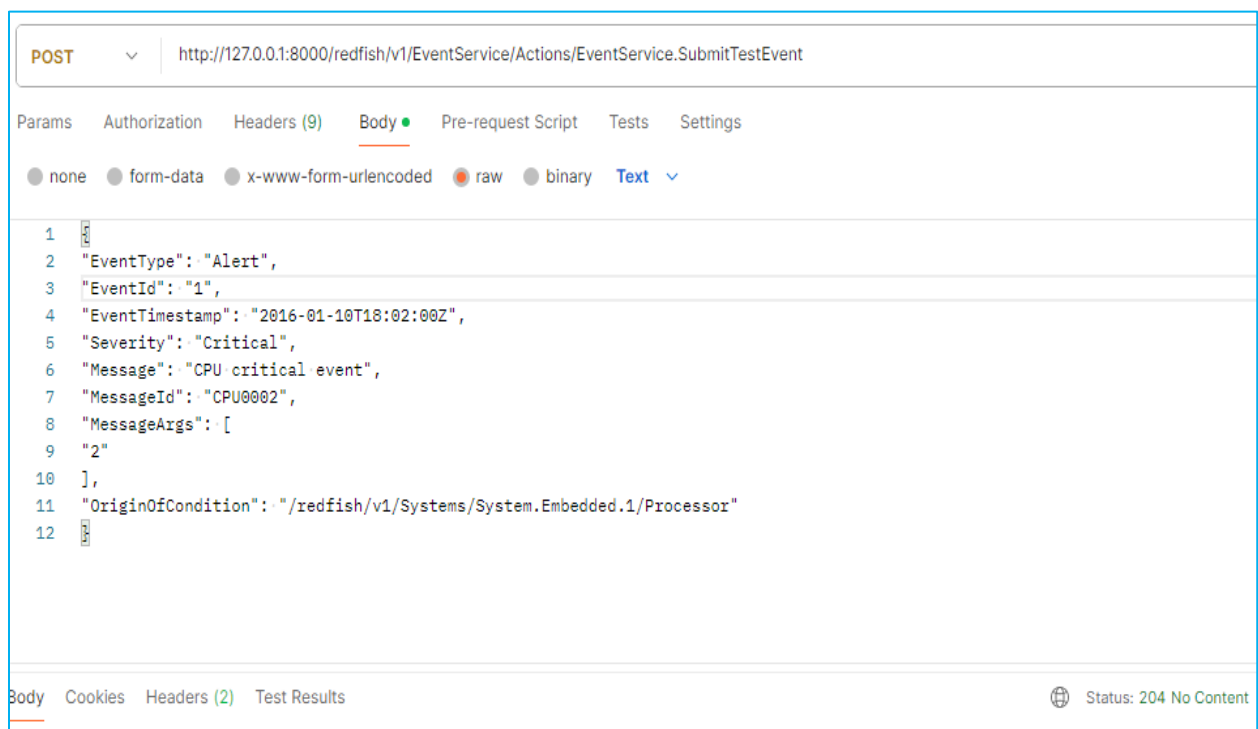


Figure 26 Submit test event

3.6 Confirm Redfish Event Listener Receives Event

Validate the Redfish Event Listener and confirm the test event you submitted is reported.

```
C:\Python310>python RedfishEventListener_v1.py
Redfish Event Listener v1.1.3
ServerIP:: http://127.0.0.1:8000
UserName:: root
Attempt 1 of /redfish/v1/
Response Time for GET to /redfish/v1/: 0.021674100076779723 seconds.
Attempt 1 of /redfish/v1/SessionService/Sessions
```

```

Response Time for POST to /redfish/v1/SessionService/Sessions:
0.009719900088384748 seconds.
Login returned code 204:
Attempt 1 of /redfish/v1/
Response Time for GET to /redfish/v1/: 0.010551500134170055 seconds.
Attempt 1 of /redfish/v1/EventService
Response Time for GET to /redfish/v1/EventService: 0.011112900217995048 seconds.
Attempt 1 of /redfish/v1/EventService/Subscriptions
Response Time for POST to /redfish/v1/EventService/Subscriptions:
0.009794200072064996 seconds.
Subscription is successful for http://127.0.0.1:8000,
/redfish/v1/EventService/Subscriptions/Member2
Service responded with invalid JSON at URI
/redfish/v1/EventService/Subscriptions

Continuing with Listener.
Listening on 0.0.0.0:80 via HTTP
Press Ctrl-C to close program
.....
Socket connected::
headers: IOrderedDict([('Host', '192.168.0.130'), ('User-Agent', 'python-
requests/2.27.1'), ('Accept-Encoding', 'gzip, deflate'), ('Accept', '*/*'),
('Connection', 'keep-alive'), ('Content-Type', 'application/json'), ('Content-
Length', '374')])

bodydata: {"@odata.type": "#Event.v1_2_1.Event", "Name": "Test Event", "Id":
"1", "Events": [{"EventType": "Alert", "EventId": "1", "EventTimestamp": "2016-
01-10T18:02:00Z", "Severity": "Critical", "Message": "CPU critical event",
"MessageId": "CPU0002", "MessageArgs": ["2"], "OriginOfCondition": {"@odata.id":
"/redfish/v1/Systems/System.Embedded.1/Processor"}}], "Context": "root"}

Server IP Address is 192.168.0.130
Server PORT number is 54633
Listener IP is 192.168.0.130

Context (root) does not match with the server (Public).
Event Counter for Host 192.168.0.130 = 1

.....

```

4 Limitations

- Authorization is not required to make any simulated Redfish calls.
- When any Redfish call is made, the Redfish mockup server session also logs request details.
- The `User-Agent` property logs the interface that invoked the Redfish request.
- The Redfish mockup server logs for the `PATCH` operations also returns the updated resource information with updated property value.
- Restart a Redfish mockup server to clear applied settings (for example, run the `delete` VD operation which removes the VD from storage inventory, restart Redfish mockup server for the VD to display again in storage inventory).
- For any issues or new iDRAC Redfish mockup client suggestions, submit your request: <https://github.com/dell/iDRAC-Redfish-Scripting/issues>.
- The `grequests` module is required to run Redfish mockup server (this module is not native to Python).
- After creating a new iDRAC user or session, GET user or session detail reports the password in clear text. This does not happen on a real iDRAC.
- An SSE streaming is not supported for Redfish events. Use Redfish Event Listener.
- Firmware update workflows (SimpleUpdate and MultipartUpload) are currently not supported by Redfish mockup server.

5 Helpful Documentation Links

- Dell Redfish API Guide: <https://developer.dell.com/apis/2978/versions/7.xx/docs/0WhatsNew.md>
- Dell iDRAC Redfish FAQ Whitepaper: <https://dl.dell.com/content/manual16004247-idrac-redfish-faq.pdf?language=en-us>
- Dell iDRAC User Interface to Redfish Mapping Whitepaper: <https://dl.dell.com/content/manual24821844-idrac9-user-interface-to-redfish-mapping.pdf?language=en-us>
- Dell iDRAC RACADM to Redfish Mapping Whitepaper: <https://dl.dell.com/content/manual9969682-idrac9-racadm-to-redfish-mapping.pdf?language=en-us>
- DMTF Redfish: <https://www.dmtf.org/standards/redfish>