3-RedfishPython

September 8, 2021

Powered by HPE DEV Team

0.0.1 Redfish REST API overview

Version 0.54

1 Using unique code to retrieve properties from different Redfish implementations

1.1 Introduction

This notebook contains a single Python program, explained and executed step by step. The goal of this lab is to retrieve the BMC MAC addresses from an OpenBMC and from an HPE iLO 5 using a single piece of code.

Please note that the account on your dedicated OpenBMC simulator has Administrator privileges, while you have Read-Only permissions on the physical shared iLO 5.

For simplicity and didactic reasons, the following code is not optimized and does not handle errors or test return codes. The simple syntax used should be easy to understand by non-Python knowledgeable students.

1.1.1 Setting the scene

The following cell imports the minimum required Python modules and defines **global variables** (credentials and IP addresses). The `redfish` module has been installed on the Jupyter server using the `pip3 install redfish` command as explained in the DMTF's Python Redfish library GitHub site.

[2]:
```
# -*- coding: utf-8 -*-

import os
import re
import sys
import json
import redfish # Module sources at: https://github.com/DMTF/
 ↪python-redfish-library/blob/master/src/redfish/rest/v1.py


##############################################################################################
#
# Variables
#
##############################################################################################

# Set Student ID number
stdid="96"
print ("You are Student" + stdid + "\n\n")

# OpenBMC and iLO common credentials
user="student"
Password='P@ssw0rd!'

# OpenBMC and iLO IP addresses with port
OpenBmcPort=44000 + int(stdid)
ObmcIP = "openbmcsimulators:" + str(OpenBmcPort)
iLO5IP = "ilo5"
```

```
You are Student96
```

1.2 Redfish session creation and Root service content

The following Python cell creates a Redfish session and saves the Root service content (`/redfish/v1`) in a variable called `service_root`.

Session details (Session Key, Token, Location…) are stored in the **`_redfishobj`** Python object. This `_redfishobj` object is automatically and silently removed at the end of the cell and the corresponding session is automatically deleted in the remote BMC.

Hence, each Jupyter cell in this notebook starts with the creation of a new session object. This may not be optimal in real life, but allows for an easy comparison of several Redfish implementations in a single notebook.

[ ]:
```
IP = ObmcIP
base_url = "https://" + IP
```

```
##############################################################################################
#
# Redfish session creation and root services listing
#
##############################################################################################
if __name__ == "__main__":
    # Set up the Redfish session object (_redfishobj)
    with redfish.redfish_client( base_url = base_url, username = user, password␣
 ↪= Password ) as _redfishobj:

        SessionKey = _redfishobj.get_session_key()
        SessionLocation = _redfishobj.get_session_location()
        print ("Session Key (Token): " + SessionKey + "\n" +
            "Session Location   : " + SessionLocation + "\n")

        # Retrieve and print the Redfish Root service content from the standard␣
 ↪location /redfish/v1/
        service_root = _redfishobj.get( "/redfish/v1/" )
        print ("Root Services:" + "\n" + json.dumps( service_root.dict,␣
 ↪indent=4 ) + "\n")
```

1.3 Managers collection listing

1.3.1 OpenBMC

As explained in the previous notebook, the standard Redfish location of a server BMC(s) is found at `/redfish/v1/Managers/{item}`, where `{item}` varies from one Redfish implementation to another.

The next cell retrieves and prints the `Managers` collection from an OpenBMC simulator. This collection has only one item named `bmc`. Run it.

[ ]:
```
IP = ObmcIP
base_url = "https://" + IP
##############################################################################################
#
# Managers service listing
#
##############################################################################################
if __name__ == "__main__":
    with redfish.redfish_client( base_url = base_url, username = user, password␣
 ↪= Password ) as _redfishobj:
        service_root = _redfishobj.get( "/redfish/v1/" )

        # Retrieve the location of the "Managers" service and its content.
        # To be perfect, we should verify that the `Managers` resource is␣
 ↪present in the ServiceRoot schema
        manager_collection = _redfishobj.get( service_root.
 ↪dict["Managers"]["@odata.id"] )
```

```
        print ( "Managers collection: " + "\n" + json.dumps( manager_collection.
↪dict, indent=4 ) + "\n" )
```

1.3.2 HPE iLO 5

The following program retrieves the `Managers` collection of an **HPE ProLiant iLO 5**.

In this example, the HPE server has only one item called `1`. Other HPE servers may have several BMCs called differently.

CONCLUSION: If you want to your Redfish programs to work against different Redfish implementations, it is mandatory to avoid assumptions on the URIs of services. You must crawl the Redfish tree and mechanically discover the desired URIs.

[ ]:
```
IP = iLO5IP
base_url = "https://" + IP
################################################################################################
#
# Managers service listing
#
################################################################################################
if __name__ == "__main__":
    with redfish.redfish_client( base_url = base_url, username = user, password␣
↪= Password ) as _redfishobj:
        service_root = _redfishobj.get( "/redfish/v1/" )

        # Retrieve the location of the "Managers" service and its content.
        # To be perfect, we should verify that the `Managers` resource is␣
↪present in the ServiceRoot schema
        manager_collection = _redfishobj.get( service_root.
↪dict["Managers"]["@odata.id"] )
        print ( "Managers location: " + "\n" + json.dumps( manager_collection.
↪dict, indent=4 ) + "\n" )
```

1.4 BMC properties and `EthernetInterfaces` collection

The next two Python cells use the exact same code to locate the BMC properties of an HPE iLO 5 first and then an OpenBMC simulator. Then, they locate and print the collection of `EthernetInterfaces` found in each BMC.

The logic of this code starts at the Root Redfish tree and then follows the desired links. The main steps are; 1) Create a Redfish session object `_redfishobj`; 2) Retrieve and save the Root service content in variable `service_root`; 3) Retrieve and save the URI of the `Managers` collection in variable `manager_collection`; 4) For each BMC URI, save its properties in variable `manager_resources`; and 5) Print those properties as well as the collection of `EthernetInterfaces`

1.4.1 OpenBMC

The OpenBMC simulator returns **two** BMC NICs under `/redfish/v1/Managers/bmc/EthernetInterfaces`.

[ ]:
```
IP = ObmcIP
base_url = "https://" + IP
##############################################################################################
#
# BMC properties and Ethernet Network Interfaces listing
#
##############################################################################################
if __name__ == "__main__":
    with redfish.redfish_client( base_url = base_url, username = user, password␣
 ↪= Password ) as _redfishobj:
        service_root = _redfishobj.get( "/redfish/v1/" )
        manager_collection = _redfishobj.get( service_root.
 ↪dict["Managers"]["@odata.id"] )

        for manager_member in manager_collection.dict["Members"]:
            manager_resources = _redfishobj.get( manager_member["@odata.id"] )
            print ( "Manager Resources" + "\n" + json.dumps(manager_resources.
 ↪dict, indent=4) + "\n" )
            print␣
 ↪("##########################################################################"␣
 ↪+ "\n")
            ethernet_network_interface_collection = _redfishobj.
 ↪get(manager_resources.dict["EthernetInterfaces"]["@odata.id"])
            print ( "Ethernet Network Interface Collection" + "\n" +
                    json.dumps(ethernet_network_interface_collection.dict,␣
 ↪indent=4) + "\n")
```

1.4.2 HPE iLO 5

The HPE iLO 5 returns **three** BMC NICs under `/redifsh/v1/Managers/1/EthernetInterfaces`.

[ ]:
```
IP = iLO5IP
base_url = "https://" + IP
##############################################################################################
#
# BMC properties and Ethernet Network Interfaces listing
#
##############################################################################################
if __name__ == "__main__":
    with redfish.redfish_client( base_url = base_url, username = user, password␣
 ↪= Password ) as _redfishobj:
        service_root = _redfishobj.get( "/redfish/v1/" )
```

```
        manager_collection = _redfishobj.get( service_root.
↪dict["Managers"]["@odata.id"] )

        for manager_member in manager_collection.dict["Members"]:
            manager_resources = _redfishobj.get( manager_member["@odata.id"] )
            print ( "Manager Resources" + "\n" + json.dumps(manager_resources.
↪dict, indent=4) + "\n" )
            print␣
↪("##########################################################################################"␣
↪+ "\n")
            ethernet_network_interface_collection = _redfishobj.
↪get(manager_resources.dict["EthernetInterfaces"]["@odata.id"])
            print ( "Ethernet Network Interface Collection" + "\n" +
                    json.dumps(ethernet_network_interface_collection.dict,␣
↪indent=4) + "\n")
```

1.5 BMC NICs MAC addresses

Using the exact same code again, we extract the MAC addresses of the two OpenBMC NICs and then the three iLO 5 NICs, regardless their different names and locations.

1.5.1 OpenBMC

[ ]:
```
IP = ObmcIP
base_url = "https://" + IP
##############################################################################################
#
# BMC NICs MAC addresses extraction
#
##############################################################################################
if __name__ == "__main__":
    with redfish.redfish_client( base_url = base_url, username = user, password␣
↪= Password ) as _redfishobj:
        # Retrieve the Redfish root services from the standard location /
↪redfish/v1/
        service_root = _redfishobj.get( "/redfish/v1/" )
        manager_collection = _redfishobj.get( service_root.
↪dict["Managers"]["@odata.id"] )

        for manager_member in manager_collection.dict["Members"]:
            manager_resources = _redfishobj.get( manager_member["@odata.id"] )
            ethernet_network_interface_collection = _redfishobj.
↪get(manager_resources.dict["EthernetInterfaces"]["@odata.id"])

            # For each ethernet network interface, get its properties and print␣
↪the location, Id and MAC Address
```

```
            for ethernet_network_interface in␣
↪ethernet_network_interface_collection.dict["Members"]:
                print (json.dumps(ethernet_network_interface))
                ethernet_network_interface_resources = _redfishobj.
↪get(ethernet_network_interface["@odata.id"])

                print (json.dumps(ethernet_network_interface_resources.
↪dict["Id"], indent=4) + ":\t" +
                        json.dumps(ethernet_network_interface_resources.
↪dict["MACAddress"], indent=4) + "\n")
```

1.5.2 HPE iLO 5

[ ]:
```
IP = iLO5IP
base_url = "https://" + IP

########################################################################################################
#
# BMC NICs MAC addresses extraction
#
########################################################################################################
if __name__ == "__main__":
    with redfish.redfish_client( base_url = base_url, username = user, password␣
↪= Password ) as _redfishobj:
        # Retrieve the Redfish root services from the standard location /
↪redfish/v1/
        service_root = _redfishobj.get( "/redfish/v1/" )
        manager_collection = _redfishobj.get( service_root.
↪dict["Managers"]["@odata.id"] )

        for manager_member in manager_collection.dict["Members"]:
            manager_resources = _redfishobj.get( manager_member["@odata.id"] )
            ethernet_network_interface_collection = _redfishobj.
↪get(manager_resources.dict["EthernetInterfaces"]["@odata.id"])

            # For each ethernet network interface, get its properties and print␣
↪the location, Id and MAC Address
            for ethernet_network_interface in␣
↪ethernet_network_interface_collection.dict["Members"]:
                print (json.dumps(ethernet_network_interface))
                ethernet_network_interface_resources = _redfishobj.
↪get(ethernet_network_interface["@odata.id"])

                print (json.dumps(ethernet_network_interface_resources.
↪dict["Id"], indent=4) + ":\t" +
```

```
                        json.dumps(ethernet_network_interface_resources.
↪dict["MACAddress"], indent=4) + "\n")
```

1.6 Congratulations !

You may now continue to the **Conclusion** notebook. Thanks you for participating to this lab.