Standards-based NIC and FC-HBA management for Dell EMC PowerEdge servers by using the iDRAC RESTful API and DMTF Redfish

June 2018

Authors

Texas Roemer, Test Principal Engineer (Dell EMC Enterprise System Test)
Paul Rubin, Sr. Product Manager (Dell EMC Server Solutions Marketing)

A Dell EMC Technical White Paper

Revisions

Date	Description
June 2018	Initial release

Contents

Executive summary

The growing scale of cloud- and web-based data center infrastructure is reshaping the requirements of IT administrators world-wide. New approaches to Systems Management are required to keep up with the growing and changing market.

The Distributed Management Task Force (DMTF) Scalable Platforms Management Forum (SPMF) has published Redfish, an open industry-standard specification and schema designed to meet the requirements of IT administrators for simple, modern, and secure management of scalable platform hardware. Dell EMC is a key contributor to the Redfish standard, acting as co-chair of the SPMF, promoting the benefits of Redfish, and working to deliver those benefits within Dell EMC industry-leading systems management solutions. Included in the Redfish 2016 standards are standardized APIs for the inventory and configuration of storage devices.

This technical white paper provides an overview with scripting examples of the Redfish APIs for Network Interface Card (NIC) and Fibre Channel Host Bus Adapter (FC-HBA) management for the12th, 13th, and 14th generation of Dell EMC PowerEdge servers, delivered by the integrated Dell Remote Access Controller (iDRAC) RESTful API.

1 Introduction

Since the inception of the x86 servers in the late 1980s, IT administrators have sought the means to efficiently manage a growing number of distributed resources. Industry suppliers have responded by developing management interface standards to support common methods of monitoring and controlling heterogeneous systems.

While management interfaces such as SNMP and IPMI have been present in data centers for the past decade, they have not been able to meet the changing requirements because of security and technical limitations.

Further, the scale of deployment has grown significantly as IT models have evolved. Today, organizations often rely on a large number of lower-cost servers with redundancy provided in the software layer, making scalable management interfaces more critical.

To meet such market requirements, a new, unifying management standard was required—the response from the industry was the creation of the Redfish systems management standard by the DTMF. Redfish is a next generation management standard by using a data model representation inside a hypermedia RESTful interface. The data model is defined in terms of a standard, machine-readable schema, with the payload of the messages expressed in JSON and the protocol using OData v4. Because it is a hypermedia API, Redfish is capable of representing a variety of implementations by using a consistent interface. It has mechanisms for discovering and managing data center resources, handling events, and managing long-lived tasks. The Redfish standard, first published in 2015, has been continuously updated, expanding the capabilities in 2016 to provide standard APIs for such mission-critical features as NIC and HBA management.

Dell EMC is enhancing its leading Systems Management capabilities with Redfish support within the iDRAC RESTful API. This technical white paper provides an overview and detailed scripted examples of the Redfish 2016 NIC and FC-HBA management APIs as implemented by the iDRAC RESTful API.

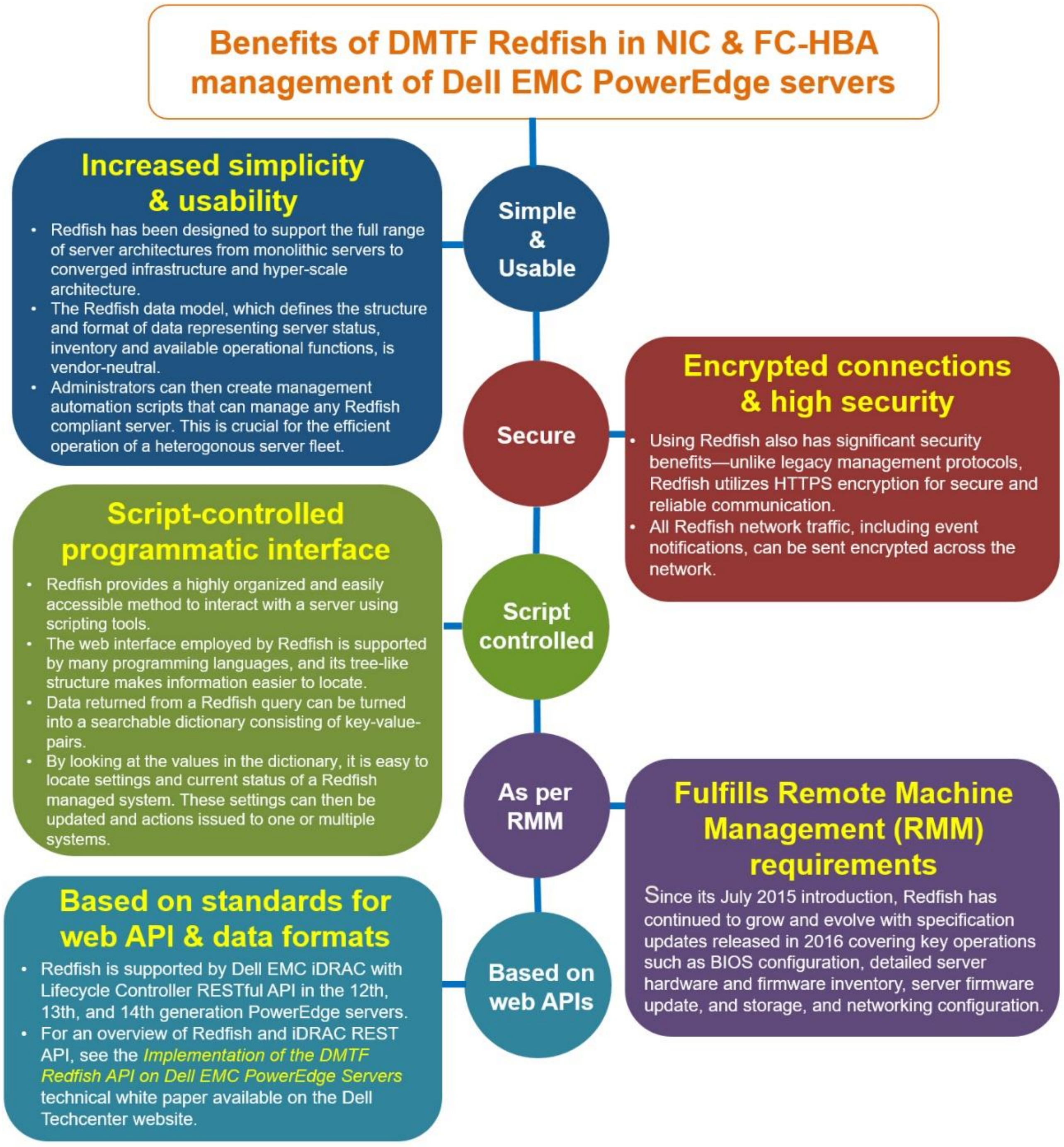
2 Introduction to iDRAC RESTful API and DMTF Redfish

There are various Out-of-Band (OOB) systems management standards available in the industry today. However, there is no single standard that can be easily used within emerging programming standards, can be readily implemented within embedded systems, and can meet the demands of today's evolving IT solution models.

New IT Solutions models have posed new demands on Systems Management Solutions to support expanded scale, higher security, and multi-vendor openness, while also aligning with modern DevOps tools and processes.

Recognizing these requirements, Dell EMC and other IT Solutions leaders within the DMTF undertook the creation of a new management interface standard. After a multi-year effort, the new standard, Redfish v1.0, was announced in July 2015.

3 Key benefits of DMTF Redfish



Benefits of DMTF Redfish in NIC & FC-HBA management of Dell EMC PowerEdge servers

Increased simplicity & usability

- Redfish has been designed to support the full range of server architectures from monolithic servers to converged infrastructure and hyper-scale architecture.
- The Redfish data model, which defines the structure and format of data representing server status, inventory and available operational functions, is vendor-neutral.
- Administrators can then create management automation scripts that can manage any Redfish compliant server. This is crucial for the efficient operation of a heterogonous server fleet.

Encrypted connections & high security

- Using Redfish also has significant security benefits—unlike legacy management protocols, Redfish utilizes HTTPS encryption for secure and reliable communication.
- All Redfish network traffic, including event notifications, can be sent encrypted across the network.

Script-controlled programmatic interface

- Redfish provides a highly organized and easily accessible method to interact with a server using scripting tools.
- The web interface employed by Redfish is supported by many programming languages, and its tree-like structure makes information easier to locate.
- Data returned from a Redfish query can be turned into a searchable dictionary consisting of key-value-pairs.
- By looking at the values in the dictionary, it is easy to locate settings and current status of a Redfish managed system. These settings can then be updated and actions issued to one or multiple systems.

Fulfills Remote Machine Management (RMM) requirements

Since its July 2015 introduction, Redfish has continued to grow and evolve with specification updates released in 2016 covering key operations such as BIOS configuration, detailed server hardware and firmware inventory, server firmware update, and storage, and networking configuration.

Based on standards for web API & data formats

- Redfish is supported by Dell EMC iDRAC with Lifecycle Controller RESTful API in the 12th, 13th, and 14th generation PowerEdge servers.
- For an overview of Redfish and iDRAC REST API, see the *Implementation of the DMTF Redfish API on Dell EMC PowerEdge Servers* technical white paper available on the Dell Techcenter website.

3.1 Redfish 2016 NIC and FC-HBA management APIs

With the advent of the Redfish 2016 standards, APIs were established for server NIC and FC-HBA management. These APIs provide the following capabilities:

- Detailed inventory of server Ethernet and Fibre Channel Host Bus Adapter devices
- Health status monitoring of NICs and FC-HBAs
- Configuration of controller settings for individual ports / channels for NICs and FC-HBAs

The balance of this white paper provides an overview of these APIs with scripted examples in Python for use with the iDRAC RESTful API. The examples scripts are available along with additional scripts for key server management use cases from the Dell EMC iDRAC RESTful API GitHub https://github.com/dell/iDRAC-Redfish-Scripting.

3.2 Inventorying PowerEdge NICs / FC-HBAs with Redfish 2016

The Redfish 2016 standard storage APIs can be used to discover network and Fibre Channel cards associated with a target PowerEdge server. The below example illustrates viewing this inventory.

- Running `GET` on URI **redfish/v1/ redfish/v1/Systems/System.Embedded.1/NetworkAdapters** will return URIs for each of the supported network device IDs detected on the target server.

Example output:

@odata.context	**"/redfish/v1/$metadata#NetworkAdapterCollection.NetworkAdapterCollection"**
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters"
@odata.type	"#NetworkAdapterCollection.NetworkAdapterCollection"
Description	"Collection Of Network Adapter"
Members	
0	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1"
1	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Slot.1"
2	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/FC.Slot.4"
Members@odata.count	3
Name	"Network Adapter Collection"

- Using each network device URI, run a GET command to return information for that device such as firmware version and model name along with individual URIs for “Network Device Functions” and “Network Ports”.

Example output:

@odata.context	"/redfish/v1/$metadata#NetworkAdapter.NetworkAdapter"
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1"
@odata.type	"#NetworkAdapter.v1_0_1.NetworkAdapter"
Controllers	
0	
ControllerCapabilities	
DataCenterBridging	
Capable	true
NPIV	
MaxDeviceLogins	0
MaxPortLogins	0
NetworkDeviceFunctionCount	4
NetworkPortCount	4
VirtualizationOffload	
SRIOV	
SRIOVVEPACapable	true
VirtualFunction	
DeviceMaxCount	0
MinAssignmentGroupSize	0
NetworkPortMaxCount	0
FirmwarePackageVersion	"09.01.05"
Links	
NetworkDeviceFunctions	
0	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkDeviceFunctions/NIC.Integrated.1-4-1"
1	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkDeviceFunctions/NIC.Integrated.1-2-1"
2	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkDeviceFunctions/NIC.Integrated.1-3-1"
3	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkDeviceFunctions/NIC.Integrated.1-1-1"
NetworkDeviceFunctions@ odata.count	4

NetworkPorts	
0	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkPorts/NIC.Integrated.1-4"
1	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkPorts/NIC.Integrated.1-2"
2	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkPorts/NIC.Integrated.1-3"
3	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkPorts/NIC.Integrated.1-1"
NetworkPorts@odata.count	4
Description	"Network Adapter View"
Id	"NIC.Integrated.1"
Manufacturer	"Dell"
Model	"BRCM 10G/GbE 2+2P 57800 rNDC"
Name	"Network Adapter View"
NetworkDeviceFunctions	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkDeviceFunctions"
NetworkPorts	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkPorts"
PartNumber	"0165T0"
SerialNumber	"CN779216CE007F"
Status	
Health	null
HealthRollup	null
State	"Enabled"

- ExeRunning GET on individual URI “Network Device Functions” will return detailed information for that specific network device. The information returned includes data such as MAC address, WWPN, iSCSI, and FCoE property settings.

  Example output below is from URI `redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkDeviceFunctions/NIC.Integrated.1-4-1`:

@Redfish.Settings	
@odata.context	"/redfish/v1/$metadata#Settings.Settings"
@odata.type	"#Settings.v1_1_0.Settings"
SettingsObject	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkDeviceFunctions/NIC.Integrated.1-4-1/Settings"
SupportedApplyTimes	

0	"OnReset"
1	"AtMaintenanceWindowStart"
2	"InMaintenanceWindowOnReset"
@odata.context	"/redfish/v1/$metadata#NetworkDeviceFunction.NetworkDeviceFunction"
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkDeviceFunctions/NIC.Integrated.1-4-1"
@odata.type	"#NetworkDeviceFunction.v1_1_0.NetworkDeviceFunction"
AssignablePhysicalPorts	
0	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkPorts/NIC.Integrated.1-4"
AssignablePhysicalPorts@ odata.count	1
Description	"NetworkDeviceFunction"
Ethernet	
MACAddress	"18:66:DA:91:9B:4E"
MTUSize	null
PermanentMACAddress	"18:66:DA:91:9B:4E"
FibreChannel	
BootTargets	
0	
LUNID	"0"
WWPN	""
FCoEActiveVLANId	1
FCoELocalVLANId	null
PermanentWWNN	"20:00:18:66:DA:91:9B:4F"
PermanentWWPN	"20:01:18:66:DA:91:9B:4F"
WWNN	"20:00:18:66:DA:91:9B:4F"
WWNSource	"ProvidedByFabric"
WWPN	"20:01:18:66:DA:91:9B:4F"
Id	"NetworkDeviceFunction"
MaxVirtualFunctions	null
Name	"NetworkDeviceFunction"
NetDevFuncCapabilities	
0	"Disabled"
1	"Ethernet"
NetDevFuncType	"Ethernet"
PhysicalPortAssignment	
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkPorts/NIC.Integrated.1-4"
Status	
Health	null
HealthRollup	null
State	"Enabled"
iSCSIBoot	
AuthenticationMethod	"None"
CHAPSecret	""

CHAPUsername	""
IPAddressType	"IPv4"
IPMaskDNSViaDHCP	true
InitiatorDefaultGateway	"0.0.0.0"
InitiatorIPAddress	"0.0.0.0"
InitiatorName	"iqn.1995-05.com.broadcom.iscsiboot"
InitiatorNetmask	"0.0.0.0"
PrimaryDNS	"0.0.0.0"
PrimaryLUN	0
PrimaryTargetIPAddress	"0.0.0.0"
PrimaryTargetName	""
PrimaryTargetTCPPort	3260
PrimaryVLANEnable	null
PrimaryVLANId	null
SecondaryDNS	"0.0.0.0"
SecondaryLUN	0
SecondaryTargetIPAddress	"0.0.0.0"
SecondaryTargetName	""
SecondaryTargetTCPPort	3260
SecondaryVLANEnable	null
SecondaryVLANId	null
TargetInfoViaDHCP	false

- Running GET on an individual URI "Network Port" returns detailed information for that specific network port. The information returned feature data such as link status, health status of the device, and various network settings.

  Example output below is from URI `redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkPorts/NIC.Integrated.1-4`:

@odata.context	"/redfish/v1/$metadata#NetworkPort.NetworkPort"
@odata.id	"/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkPorts/NIC.Integrated.1-4"
@odata.type	"#NetworkPort.v1_1_0.NetworkPort"
ActiveLinkTechnology	"Ethernet"
AssociatedNetworkAddresses	
0	"18:66:DA:91:9B:4E"
Description	"Network Port View"
EEEEnabled	null
FlowControlConfiguration	"None"
FlowControlStatus	"None"

Id	"NIC.Integrated.1-4"
LinkStatus	"Down"
Name	"Network Port View"
NetDevFuncMaxBWAlloc	[]
NetDevFuncMinBWAlloc	[]
PhysicalPortNumber	"4"
Status	
Health	null
HealthRollup	null
State	"Enabled"
SupportedEthernetCapabilities	
0	"WakeOnLAN"
SupportedLinkCapabilities	
0	
LinkNetworkTechnology	"Ethernet"
LinkSpeedMbps	0
WakeOnLANEnabled	false

3.3 Configuring network device properties by using Redfish 2016

To configure network device properties such as enabling iSCSI and FCoE, run a PATCH command by using the target URI for a selected network port `redfish/v1/Systems/System.Embedded.1/NetworkAdapters/ {network device id}/NetworkDeviceFunctions/{network port}/Settings` with a payload passing in the selected properties. Running the PATCH command sets the property to a pending value. To apply the change, execute a PATCH command to create a configuration job by using "@Redfish.SettingsApplyTime".

Examples of the URI and payload dictionary for PATCH command to set pending property values:

```
/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkDeviceFunctions/NIC.Integrated.1-4-1/Settings

“payload={"iSCSIBoot":{"InitiatorIPAddress":"192.168.0.120","InitiatorNetmask":"255.255.255.0"},"FibreChannel":{"FCoELocalVLANId":100}}”
```

Examples of the URI and payload dictionary for PATCH command to create the configuration job to apply the changes.

Note: URI for creating the configuration job will be the same one that was used for PATCH command to set property pending values.

```
/redfish/v1/Systems/System.Embedded.1/NetworkAdapters/NIC.Integrated.1/NetworkDeviceFunctions/NIC.Integrated.1-4-1/Settings

payload = {"@Redfish.SettingsApplyTime":{"ApplyTime":"OnReset"}}
```

Note: This PATCH command will create and schedule the configuration job but it WILL NOT reboot the server to apply the change. When ready, use POST a command with ACTION "ComputerSystem.Reset" to reboot the server; this will cause iDRAC with Lifecycle Controller to execute the job and apply the changes.

The example below illustrates using the `Python script SetNetworkDevicePropertiesRedfish.py` from the Dell EMC Redfish Scripting GitHub repository to inventory the server's network devices, display their current settings, and then set the iSCSI boot network properties.

Note: A best practice is to first run the script with the `-h` option to view the help text which explains the supported parameters and values and provides usage examples.

1. Inventory the network devices on the target server:

```
C:\Python27>SetNetworkDevicePropertiesRedfish.py -ip 192.168.0.120 -u root -p
calvin -n y
- Network device ID(s) detected -
NIC.Integrated.1
NIC.Slot.1
FC.Slot.4

- Network port ID(s) detected for NIC.Integrated.1 -
NIC.Integrated.1-4-1
NIC.Integrated.1-2-1
NIC.Integrated.1-3-1
NIC.Integrated.1-1-1

- Network port ID(s) detected for NIC.Slot.1 -
NIC.Slot.1-4-1
NIC.Slot.1-4-2
NIC.Slot.1-4-3
NIC.Slot.1-4-4
NIC.Slot.1-2-1
NIC.Slot.1-2-2
NIC.Slot.1-2-3
NIC.Slot.1-2-4
NIC.Slot.1-3-1
NIC.Slot.1-3-2
NIC.Slot.1-3-3
NIC.Slot.1-3-4
NIC.Slot.1-1-1
NIC.Slot.1-1-2
NIC.Slot.1-1-3
NIC.Slot.1-1-4

- Network port ID(s) detected for FC.Slot.4 -
FC.Slot.4-1
```

2. Get current property settings for a specific network device port. This example will use "NIC.Integrated.1-1-1":

```
C:\Python27>SetNetworkDevicePropertiesRedfish.py -ip 192.168.0.120 -u root -p
calvin -a NIC.Integrated.1-1-1

- Properties for network device NIC.Integrated.1-1-1 -

 - iSCSIBoot Properties -

InitiatorDefaultGateway: 0.0.0.0
SecondaryTargetName:
CHAPUsername:
PrimaryLUN: 0
SecondaryVLANEnable: None
SecondaryVLANId: None
SecondaryDNS: 0.0.0.0
InitiatorNetmask: 0.0.0.0
IPMaskDNSViaDHCP: True
PrimaryTargetName:
PrimaryDNS: 0.0.0.0
PrimaryTargetTCPPort: 3260
SecondaryTargetTCPPort: 3260
IPAddressType: IPv4
TargetInfoViaDHCP: False
InitiatorName: iqn.1995-05.com.broadcom.iscsiboot
InitiatorIPAddress: 0.0.0.0
SecondaryTargetIPAddress: 0.0.0.0
SecondaryLUN: 0
CHAPSecret:
PrimaryVLANId: None
AuthenticationMethod: None
PrimaryTargetIPAddress: 0.0.0.0
PrimaryVLANEnable: None

 - FibreChannel Properties -

WWNN: 20:00:18:66:DA:91:9B:49
PermanentWWNN: 20:00:18:66:DA:91:9B:49
PermanentWWPN: 20:01:18:66:DA:91:9B:49
FCoELocalVLANId: None
FCoEActiveVLANId: 1
WWNSource: ProvidedByFabric
WWPN: 20:01:18:66:DA:91:9B:49
BootTargets: [{u'WWPN': u'00:00:00:00:00:00:00:00', u'LUNID': u'0'}
```

Note: Some device properties are read-only while other properties are read-write. To determine the properties that can be modified under different conditions, see documentation of the selected network controller interface. An example is iSCSI boot properties—if iSCSI is being configured by using DHCP then iSCSI boot property settings cannot be modified unless DHCP usage is disabled.

3. Before setting properties, use the script to generate an “ini” file which will be used by the script to set properties to new values. The generated “ini” file will be output in the required dictionary structure to set the network properties.

```
C:\Python27>SetNetworkDevicePropertiesRedfish.py -ip 192.168.0.120 -u root
-p calvin -g y
- WARNING, "set network_properties.ini" file created. This file contains
payload dictionary which will be used to set properties
.
Modify the payload dictionary passing in property names and values for the
correct group.
Example of modified dictionary:
{"iSCSIBoot":{"InitiatorIPAddress":"192.168.0.120","InitiatorNetmask":"255
.255.255.0"},"FibreChannel":{"FCoELocalVLANId":100}}
```

4. This workflow will be used to configure the iSCSI Boot property “InitiatorIPAddress”. The script must pass in this key along with a string value for the nested dictionary “iSCSIBoot”.

   Example of the modified “ini” file:

```
{"FibreChannel": {}, "iSCSIBoot": {"InitiatorIPAddress":"192.168.0.140"}}
```

   Note: For the property group **FibreChannel**, enter either an empty dictionary or remove it from the “ini” file.

5. After the “ini” file is modified as necessary, apply the configuration changes to be applied immediately.

   Note: When applying network property changes, the configuration job can be scheduled for a maintenance window occurring at a time in the future. For more information, see `@Redfish.SettingsApplyTime` in the Redfish schema for more information.

```
C:\Python27>SetNetworkDevicePropertiesRedfish.py -ip 192.168.0.120 -u root -p
calvin -s NIC.Integrated.1-1-1 -r n
- WARNING, setting properties for iSCSIBoot group:
Property Name: InitiatorIPAddress, Pending New Value: 192.168.0.140

- PASS: PATCH command passed to set property pending value, status code 200
returned
- PASS: PATCH command passed to create next reboot config job, status code
202 returned
- PASS, JID_277824009347 next reboot config jid successfully created
- WARNING: JobStatus not scheduled, current status is: New
- WARNING, config job marked as scheduled, system will now reboot to apply
configuration changes
- PASS, Command passed to power OFF server, code return is 204
- PASS, Command passed to power ON server, code return is 204

- WARNING, JobStatus not completed, current status is: "Task successfully
scheduled.", percent completion is: "0"
<…>
- WARNING, JobStatus not completed, current status is: "Job in progress.",
percent completion is: "34"
<Output edited for brevity>

--- PASS, Final Detailed Job Status Results ---

JobState: Completed
Description: Job Instance
CompletionTime: 2018-05-31T11:10:31
PercentComplete: 100
StartTime: TIME_NOW
MessageId: PR19
Message: Job completed successfully.
EndTime: TIME_NA
Id: JID_277824009347
JobType: NICConfiguration
Name: ConfigNIC:NIC.Integrated.1-1-1

- JID_277824009347 job execution time: 0:10:26
```

6. Verify the new property settings to ensure that “InitiatorIPAddress” is set to 192.168.0.140:

```
C:\Python27>SetNetworkDevicePropertiesRedfish.py -ip 192.168.0.120 -u root -p
calvin -a NIC.Integrated.1-1-1
- Properties for network device NIC.Integrated.1-1-1 –

 - iSCSIBoot Properties -
InitiatorDefaultGateway: 0.0.0.0
SecondaryTargetName:
CHAPUsername:
PrimaryLUN: 0
SecondaryVLANEnable: None
SecondaryVLANId: None
SecondaryDNS: 0.0.0.0
InitiatorNetmask: 0.0.0.0
IPMaskDNSViaDHCP: False
PrimaryTargetName:
PrimaryDNS: 0.0.0.0
PrimaryTargetTCPPort: 3260
SecondaryTargetTCPPort: 3260
IPAddressType: IPv4
TargetInfoViaDHCP: False
InitiatorName: iqn.1995-05.com.broadcom.iscsiboot
InitiatorIPAddress: 192.168.0.140
SecondaryTargetIPAddress: 0.0.0.0
SecondaryLUN: 0
CHAPSecret:
```

```
PrimaryVLANId: None
AuthenticationMethod: None
PrimaryTargetIPAddress: 0.0.0.0
PrimaryVLANEnable: None

 - FibreChannel Properties -

WWNN: 20:00:18:66:DA:91:9B:49
PermanentWWNN: 20:00:18:66:DA:91:9B:49
PermanentWWPN: 20:01:18:66:DA:91:9B:49
FCoELocalVLANId: None
FCoEActiveVLANId: 1
WWNSource: ProvidedByFabric
WWPN: 20:01:18:66:DA:91:9B:49
BootTargets: [{u'WWPN': u'00:00:00:00:00:00:00:00', u'LUNID': u'0'}]
```

3.4 Limitations when using Redfish 2016 standard network APIs for network configuration

There are several limitations when using Redfish 2016 standard APIs for network configurations:

- When operating upon the network properties for a port, use the Redfish schema (NetworkDeviceFunction) to determine if the property is read-only or read-write. The schema does not provide information about dependencies of a given property on other properties. Dell EMC provides attribute registries on delltechcenter.com/idrac which describes the available network properties, their read/write capabilities, and any dependencies on other properties.
- Redfish 2016 standard network APIs support setting a limited number of network properties. For example, the standard APIs do not support enabling VLAN mode, setting the VLAN port, or support advanced configuration such as NIC partitioning. Such detailed configuration can be implemented by using the Dell EMC Server Configuration Profile (SCP) feature. For more details about using SCP features for network configuration see the RESTful Server Configuration with iDRAC REST API white paper available on the Dell EMC Techcenter.

4 Summary

The DMTF Redfish standard is emerging as a key new tool for efficient, scalable, and secure server management. Utilizing an industry-standard interface and data format, Redfish supports rapid development of automation for one-to-many server management. System administrators and IT developers will appreciate Redfish features that can increase efficiency, lower costs and boost productivity across their organizations.

Using the iDRAC RESTful API with Redfish 2016 support, administrators can script the automation of server NIC and HBA functions including detailed inventory, health monitoring and the configuration of controllers and individual ports and channels.

Dell EMC is a committed leader in the development and implementation of open, industry standards. Supporting Redfish within the Dell EMC iDRAC with Lifecycle Controller further enhances the manageability of PowerEdge servers, providing another powerful tool to help IT administrators reduce complexity while increasing the efficiency of their operations.

5 Additional information

- For more information on iDRAC9 and 14G BIOS, visit the BIOS section of the iDRAC9 white paper library on Dell Techcenter http://delltechcenter.com/idrac
- DMTF white papers, Redfish Schemas, specifications, webinars and work-in-progress documents https://www.dmtf.org/standards/redfish
- The Redfish standard specification is available from the DMTF website
- http://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.0.1.pdf
- The iDRAC with Lifecycle Controller home page on Dell TechCenter provides access to product documents, technical white papers, how-to videos and more
- http://en.community.dell.com/techcenter/systems-management/w/wiki/3204
- GitHub repository for iDRAC REST API with Redfish support for PowerShell and Python https://github.com/dell/iDRAC-Redfish-Scripting