Standards-based storage management for Dell EMC PowerEdge servers 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 needed 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 needs 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 storage 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 server 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.

Also, 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 necessary—the response from the industry was the creation of the Redfish systems management standard by the DTMF. Redfish is a next generation management standard 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 storage 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 storage management APIs as implemented by the iDRAC RESTful API.

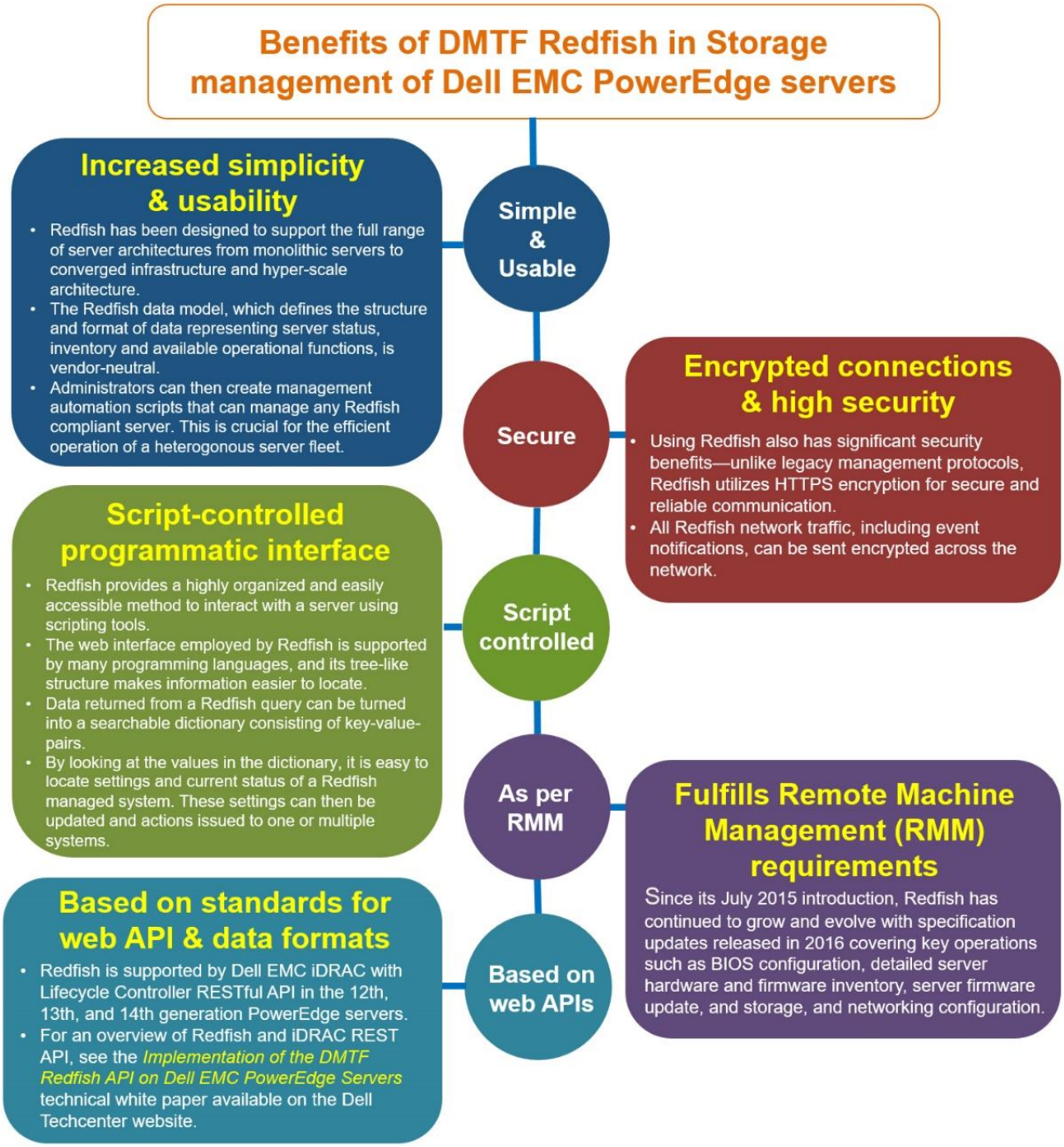
2 iDRAC RESTful API and DMTF Redfish

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 placed 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 needs, 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.

2.1 Key benefits of DMTF Redfish



3 Redfish 2016 Storage Management APIs

With the advent of the Redfish 2016 standards, APIs were established for server storage management. These APIs provide the following capabilities:

- Detailed inventory of server storage controllers, enclosures, physical drive and virtual disks
- Health status monitoring of controllers, enclosures and drives
- Physical drive operations such as drive erase
- Controller operations such as create/delete virtual disk, virtual disk initialization/consistency check
- External enclosure operations such as setting the asset tag

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 example 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.

3.1 Inventorying PowerEdge storage with Redfish 2016

The Redfish 2016 standard storage APIs can be used to discover the storage controllers, storage enclosures, and physical and virtual drives (VDs) associated with a target PowerEdge server. The below example illustrates viewing this inventory.

1. Running GET on URI `redfish/v1/Systems/System.Embedded.1/Storage` will return URIs for each of the supported storage controllers detected on the target server.

Example output:

```
@odata.context "/redfish/v1/$metadata#StorageCollection.StorageCollection"
@odata.id      "/redfish/v1/Systems/System.Embedded.1/Storage"
@odata.type    "#StorageCollection.StorageCollection"
Description    "Collection Of Storage entities"
Members
0: @odata.id:  "/redfish/v1/Systems/System.Embedded.1/Storage/RAID.Integrated.1-1"
1: @odata.id:  "/redfish/v1/Systems/System.Embedded.1/Storage/CPU.1"
2: @odata.id:  "/redfish/v1/Systems/System.Embedded.1/Storage/AHCI.Slot.1-1"
3: @odata.id:  "/redfish/v1/Systems/System.Embedded.1/Storage/AHCI.Embedded.2-1"
4: @odata.id:  "/redfish/v1/Systems/System.Embedded.1/Storage/AHCI.Embedded.1-1"
5: @odata.id:  "/redfish/v1/Systems/System.Embedded.1/Storage/PCIeSSD.Slot.2-C"
Members@odata.count: 6
Name:    "Storage Collection"
```

2. Using each controller URIs, run a GET command to return detailed information for that controller along with the URIs for the drives and volumes associated with the controller.
3. Run GET on each drive URI to return detailed information for that specific drive. Volume URIs will return virtual disk URIs if VDs are detected on the server.
4. Run GET on the VD URI to get detailed information for that specific VD.

Example output:

```
@odata.context "/redfish/v1/$metadata#Storage.Storage"
@odata.id      "/redfish/v1/Systems/System.Embedded.1/Storage/RAID.Integrated.1-
1"
@odata.type    "#Storage.v1_3_0.Storage"
Description    "PERC H330 Mini"
Drives
0  @odata.id
   "/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.0:Enclosure.In
ternal.0-1:RAID.Integrated.1-1"
1  @odata.id
   "/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.1:Enclosure.In
ternal.0-1:RAID.Integrated.1-1"
2  @odata.id
   "/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.3:Enclosure.In
ternal.0-1:RAID.Integrated.1-1"
Drives@odata.count   3

Id "RAID.Integrated.1-1"
Links
Enclosures
0  @odata.id   "/redfish/v1/Chassis/Enclosure.Internal.0-1:RAID.Integrated.1-1"
Enclosures@odata.count     1

Name     "PERC H330 Mini"
Status
Health   null
HealthRollup   null
State    "Enabled"

StorageControllers
0  @odata.id
   "/redfish/v1/Systems/System.Embedded.1/StorageControllers/RAID.Integrated.1-
1"
FirmwareVersion      "25.5.4.0006"
Identifiers
0
DurableName    "51866DA0B2425700"
DurableNameFormat    "NAA"
Links    {}
Manufacturer   "DELL"
MemberId "RAID.Integrated.1-1"
Model    "PERC H330 Mini"
Name     "PERC H330 Mini"
SpeedGbps      12
Status
Health   null
HealthRollup   null
State    "Enabled"

SupportedControllerProtocols
0  "PCIe"
```

```
SupportedDeviceProtocols
0   "SAS"
1   "SATA"

StorageControllers@odata.count   1
Volumes
@odata.id      "/redfish/v1/Systems/System.Embedded.1/Storage/RAID.Integrated.1-
1/Volumes"
```

3.2 Creating a virtual drive (VD) by using Redfish 2016

To create a VD, run a POST command by using the target storage controller URI "redfish/v1/Systems/System.Embedded.1/Storage/{selected storage controller instance ID}/Volumes" with payload required parameters **VolumeType** and **Drives**.

Note: Redfish 2016 schema Storage section will list all the required / optional parameters with supported values for creating a VD.

The following example shows how to use the controller URI with a payload dictionary and POST command to create a VD by using Python:

```
url = 'https://%s/redfish/v1/Systems/System.Embedded.1/Storage/RAID.Slot.6-
1/Volumes' % idrac_ip

payload = {"VolumeType":"Mirrored","Drives":[{"@odata.id":
"/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.0:Enclosure.Inter
nal.0-1:RAID.Slot.6-1"},{"@odata.id":
"/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.1:Enclosure.Inter
nal.0-1:RAID.Slot.6-1"}]}

response = requests.post(url, data=json.dumps(payload), headers=headers,
verify=False,auth=(idrac_username,idrac_password))
```

The example below illustrates using the Python script "CreateVirtualDiskRedfish.py" from the Dell EMC Redfish Scripting Github repository to create a 10 GB RAID-0 VD.

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. Check for the storage controllers detected on the target server.

```
C:\Python27>CreateVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p calvin -c y
- Server controller(s) detected -
RAID.Slot.6-1
PCIeExtender.Slot.1
AHCI.Embedded.1-1
AHCI.Embedded.2-1
PCIeExtender.Slot.7
PCIeExtender.Slot.4
```

2. View storage controller RAID.Slot.6-1 for the detected drives attached to the controller.

```
C:\Python27>CreateVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p calvin -d
RAID.Slot.6-1
- Drive(s) detected for RAID.Slot.6-1 –
Disk.Bay.7:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.9:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.0:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.1:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.2:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.3:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.4:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.5:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.6:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.8:Enclosure.Internal.0-1:RAID.Slot.6-1
```

3. Create a RAID 0 VD composed of two physical drives, 10 GB each in size.

```
C:\Python27>CreateVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p calvin -V y
-C RAID.Slot.6-1 -D Disk.Bay.0:Enclosure.Internal.0-1:RAID.Slot.6-
1,Disk.Bay.1:Enclosure.Internal.0-1:RAID.Slot.6-1 -R 0 --size 10737418240

- PASS: POST command passed to create "NonRedundant" virtual disk, status code
202 returned

- PASS, "realtime" JID_264172337521 jid successfully created for create virtual
disk
- WARNING, JobStatus not completed, current status is: "New", precent completion
is: "0"
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "1"
<Output edited for brevity…>
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "99"

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

JobState: Completed
Description: Job Instance
CompletionTime: 2018-05-15T15:48:39
PercentComplete: 100
StartTime: TIME_NOW
MessageId: PR19
Message: Job completed successfully.
EndTime: TIME_NA
Id: JID_264172337521
JobType: RealTimeNoRebootConfiguration
Name: Config:RAID:RAID.Slot.6-1
```

4. Validate that the RAID 0 VD was created and check the details.

```
C:\Python27>CreateVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p calvin -v
RAID.Slot.6-1
- Volume(s) detected for RAID.Slot.6-1 controller -
Disk.Virtual.0:RAID.Slot.6-1, Volume type: NonRedundant

C:\Python27>CreateVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p calvin -vv
RAID.Slot.6-1
- Volume(s) detected for RAID.Slot.6-1 controller -
Disk.Virtual.0:RAID.Slot.6-1
 - Detailed Volume information for Disk.Virtual.0:RAID.Slot.6-1 -
@odata.type: #Volume.v1_0_3.Volume
Operations: []
Status: {u'HealthRollup': u'OK', u'State': u'Enabled', u'Health': u'OK'}
Description: Virtual Disk 0
Links: {u'Drives': [{u'@odata.id':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.0:Enclosure.Inte
rnal.0-1:RAID.Slot.6-1'}, {u'@odata.id':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.1:Enclosure.Inte
rnal.0-1:RAID.Slot.6-1'}], u'Drives@odata.count': 2}
BlockSizeBytes: 512
Encrypted: False
OptimumIOSizeBytes: 262144
Identifiers: []
VolumeType: NonRedundant
@odata.id:
/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot.6
-1
Actions: {u'#Volume.CheckConsistency': {u'target':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot
.6-1/Actions/Volume.CheckConsistency'}, u'#Volume.Initialize':
{u'InitializeType@Redfish.AllowableValues': [u'Fast', u'Slow'], u'target':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot
.6-1/Actions/Volume.Initialize'}}
EncryptionTypes: [u'NativeDriveEncryption']
CapacityBytes: 10737418240
@odata.context: /redfish/v1/$metadata#Volume.Volume
Id: Disk.Virtual.0:RAID.Slot.6-1
@Redfish.Settings: {u'@odata.type': u'#Settings.v1_1_0.Settings',
u'SettingsObject': {u'@odata.id':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot
.6-1/Settings'}, u'@odata.context': u'/redfish/v1/$metadata#Settings.Settings',
u'SupportedApplyTimes': [u'Immediate', u'OnReset', u'AtMaintenanceWindowStart',
u'InMaintenanceWindowOnReset']}
Name: Virtual Disk 0
```

3.3 Deleting a virtual drive (VD) by using Redfish 2016

VDs can also be deleted by using Redfish 2016 standard storage API. By using a selected VD URI of form “redfish/v1/Systems/System.Embedded.1/Storage/Volumes/{virtual disk instance id}”, with no payload required, use a DELETE command to delete the VD.

Here is an example, taken from the “DeleteVirtualDiskRedfish.py” script, of the form of the URI and DELETE command for deleting VD from Python GitHub script.

```
url = 'https://%s/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/%s' %
(idrac_ip, virtual_disk)
    headers = {'content-type': 'application/json'}
    response = requests.delete(url, headers=headers,
verify=False,auth=(idrac_username,idrac_password))
```

The following example uses the “DeleteVirtualDiskRedfish.py” script from the Dell EMC Redfish Scripting Github repository to delete a RAID 0 VD:

1. Check for VDs detected on the target server.

```
C:\Python27>DeleteVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p calvin -v
RAID.Slot.6-1
- Supported virtual disk(s) detected to delete for controller RAID.Slot.6-1 -
Disk.Virtual.0:RAID.Slot.6-1, Volume Type: NonRedundant
```

2. Using the VD instance ID, perform the VD delete operation.

```
C:\Python27>DeleteVirtualDiskRedfish.py –ip 192.168.0.130 -u root -p calvin -D
Disk.Virtual.0:RAID.Slot.6-1

- PASS: DELETE command passed to delete "Disk.Virtual.0:RAID.Slot.6-1" virtual
disk, status code 202 returned
- PASS, "realtime" JID_264179257876 jid successfully created for delete virtual
disk

- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "1"
<Output edited for brevity>
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "99"

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

JobState: Completed
Description: Job Instance
CompletionTime: 2018-05-15T15:59:58
PercentComplete: 100
StartTime: TIME_NOW
MessageId: PR19
Message: Job completed successfully.
EndTime: TIME_NA
Id: JID_264179257876
```

```
JobType: RealTimeNoRebootConfiguration
Name: Config:RAID:RAID.Slot.6-1
```

3. Ensure that the VD no longer exists for the target controller.

```
C:\Python27>DeleteVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p calvin -v
RAID.Slot.6-1
- WARNING, no volume(s) detected for RAID.Slot.6-1
```

3.4 Initialize virtual drive (VD) by using Redfish 2016

After a VD is created, the Redfish 2016 storage APIs are used to initialize the VD for usage. Initialization is performed by using a POST command with a virtual disk URI "redfish/v1/Systems/System.Embedded.1/Storage/Volumes/{virtual disk instance ID}/Actions/ Volume.Initialize" and the required payload dictionary.

Here is an example from the "InitializeVirtualDiskRedfish.py" script in the Dell EMC Redfish Scripting GitHub repository of the URI, payload and POST command to initialize virtual disks:

```
url =
'https://%s/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/%s/Actions/Volu
me.Initialize' % (idrac_ip, virtual_disk)
    payload={"InitializeType":init_type}
    headers = {'content-type': 'application/json'}
    response = requests.post(url, data=json.dumps(payload), headers=headers,
verify=False,auth=(idrac_username,idrac_password))
```

The example below uses the "InitializeVirtualDiskRedfish.py" script to initialize a RAID 1 VD.

1. Check the virtual disk(s) detected on the target server for the RAID 1 virtual disk:

```
C:\Python27>InitializeVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p calvin
-v RAID.Slot.6-1
- Virtual disk(s) detected for controller RAID.Slot.6-1 -
Disk.Virtual.0:RAID.Slot.6-1, Volume Type: Mirrored
```

2. Run a "Slow" initialization on the RAID 1 VD, passing in the VD instance ID:

```
C:\Python27>InitializeVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p calvin
-V Disk.Virtual.0:RAID.Slot.6-1 --init Slow
- PASS: POST command passed to Slow initialize "Disk.Virtual.0:RAID.Slot.6-1"
virtual disk, status code 202 returned
- PASS, "realtime" JID_265044041574 jid successfully created for initialize
virtual disk
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "1"
<Output edited for brevity>
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "99"
--- PASS, Final Detailed Job Status Results ---
```

```
JobState: Completed
Description: Job Instance
CompletionTime: 2018-05-16T16:01:15
PercentComplete: 100
StartTime: TIME_NOW
MessageId: PR19
Message: Job completed successfully.
EndTime: TIME_NA
Id: JID_265044041574
JobType: RealTimeNoRebootConfiguration
Name: Config:RAID:RAID.Slot.6-1
```

3. Check the VD details to ensure that the **Operations** property reports initialization is in progress.

```
C:\Python27>InitializeVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p calvin
-vv RAID.Slot.6-1
- Volume(s) detected for RAID.Slot.6-1 controller -
Disk.Virtual.0:RAID.Slot.6-1
 - Detailed Volume information for Disk.Virtual.0:RAID.Slot.6-1 -

@odata.type: #Volume.v1_0_3.Volume
Operations: [{u'AssociatedTask': None, u'OperationName': u'Initialization',
u'PercentageComplete': 37}]
Status: {u'HealthRollup': u'OK', u'State': u'Enabled', u'Health': u'OK'}
Description: Virtual Disk 0
Links: {u'Drives': [{u'@odata.id':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.0:Enclosure.Inte
rnal.0-1:RAID.Slot.6-1'}, {u'@odata.id':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.1:Enclosure.Inte
rnal.0-1:RAID.Slot.6-1'}], u'Drives@odata.count': 2}
BlockSizeBytes: 512
Encrypted: False
OptimumIOSizeBytes: 262144
Identifiers: []
VolumeType: Mirrored
@odata.id:
/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot.6
-1
Actions: {u'#Volume.CheckConsistency': {u'target':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot
.6-1/Actions/Volume.CheckConsistency'}, u'#Volume.Initialize':
{u'InitializeType@Redfish.AllowableValues': [u'Fast', u'Slow'], u'target':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot
.6-1/Actions/Volume.Initialize'}}
EncryptionTypes: [u'NativeDriveEncryption']
CapacityBytes: 32212254720
@odata.context: /redfish/v1/$metadata#Volume.Volume
Id: Disk.Virtual.0:RAID.Slot.6-1
@Redfish.Settings: {u'@odata.type': u'#Settings.v1_1_0.Settings',
u'SettingsObject': {u'@odata.id':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot
.6-1/Settings'}, u'@odata.context': u'/redfish/v1/$metadata#Settings.Settings',
```

```
u'SupportedApplyTimes': [u'Immediate', u'OnReset', u'AtMaintenanceWindowStart',
u'InMaintenanceWindowOnReset']}
Name: Virtual Disk 0
```

3.5 Check virtual drive (VD) consistency by using Redfish 2016

After a VD is established, it can be checked for consistency with the Redfish 2016 storage APIs. Consistency checking is done by using a POST command on the VD URI "redfish/v1/Systems/System.Embedded.1/Storage/Volumes/{virtual disk instance ID}/Actions/Volume.CheckConsistency" with no payload dictionary.

Here is an example from the "CheckConsistencyVirtualDiskRedfish.py" script in the Dell EMC Redfish Scripting GitHub repository of the URI and POST command to check consistency on a VD:

```
url =
'https://%s/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/%s/Actions/Volu
me.CheckConsistency' % (idrac_ip, virtual_disk)
    headers = {'content-type': 'application/json'}
    response = requests.post(url, headers=headers,
verify=False,auth=(idrac_username,idrac_password))
```

The following example uses "CheckConsistencyVirtualDiskRedfish.py" to check the consistency of a RAID 1 VD:

1. View the VDs detected on the target server for the RAID 1 VD:

```
C:\Python27>CheckConsistencyVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p
calvin -v RAID.Slot.6-1
- Virtual disk(s) detected for controller RAID.Slot.6-1 -
Disk.Virtual.0:RAID.Slot.6-1, Volume Type: Mirrored
```

2. Run check consistency on the RAID 1 VD by entering the VD instance ID:

```
C:\Python27>CheckConsistencyVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p
calvin -cc Disk.Virtual.0:RAID.Slot.6-1
- PASS: POST command passed to check consistency "Disk.Virtual.0:RAID.Slot.6-1"
virtual disk, status code 202 returned
- PASS, "realtime" JID_265050309108 jid successfully created for check
consistency virtual disk

- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "1"
<Output edited for brevity>
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "99"
--- PASS, Final Detailed Job Status Results ---
JobState: Completed
Description: Job Instance
CompletionTime: 2018-05-16T16:11:41
PercentComplete: 100
StartTime: TIME_NOW
```

```
MessageId: PR19
Message: Job completed successfully.
EndTime: TIME_NA
Id: JID_265050309108
JobType: RealTimeNoRebootConfiguration
Name: Config:RAID:RAID.Slot.6-1
```

3. Check the VD details and ensure that the **Operations** property reports check consistency is in progress:

```
C:\Python27>CheckConsistencyVirtualDiskRedfish.py -ip 192.168.0.130 -u root -p
calvin -vv RAID.Slot.6-1
- Volume(s) detected for RAID.Slot.6-1 controller -
Disk.Virtual.0:RAID.Slot.6-1
 - Detailed Volume information for Disk.Virtual.0:RAID.Slot.6-1 -
@odata.type: #Volume.v1_0_3.Volume
Operations: [{u'AssociatedTask': None, u'OperationName': u'Check Consistency',
u'PercentageComplete': 55}]
Status: {u'HealthRollup': u'OK', u'State': u'Enabled', u'Health': u'OK'}
Description: Virtual Disk 0
Links: {u'Drives': [{u'@odata.id':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.0:Enclosure.Inte
rnal.0-1:RAID.Slot.6-1'}, {u'@odata.id':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Drives/Disk.Bay.1:Enclosure.Inte
rnal.0-1:RAID.Slot.6-1'}], u'Drives@odata.count': 2}
BlockSizeBytes: 512
Encrypted: False
OptimumIOSizeBytes: 262144
Identifiers: []
VolumeType: Mirrored
@odata.id:
/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot.6
-1
Actions: {u'#Volume.CheckConsistency': {u'target':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot
.6-1/Actions/Volume.CheckConsistency'}, u'#Volume.Initialize':
{u'InitializeType@Redfish.AllowableValues': [u'Fast', u'Slow'], u'target':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot
.6-1/Actions/Volume.Initialize'}}
EncryptionTypes: [u'NativeDriveEncryption']
CapacityBytes: 32212254720
@odata.context: /redfish/v1/$metadata#Volume.Volume
Id: Disk.Virtual.0:RAID.Slot.6-1
@Redfish.Settings: {u'@odata.type': u'#Settings.v1_1_0.Settings',
u'SettingsObject': {u'@odata.id':
u'/redfish/v1/Systems/System.Embedded.1/Storage/Volumes/Disk.Virtual.0:RAID.Slot
.6-1/Settings'}, u'@odata.context': u'/redfish/v1/$metadata#Settings.Settings',
u'SupportedApplyTimes': [u'Immediate', u'OnReset', u'AtMaintenanceWindowStart',
u'InMaintenanceWindowOnReset']}
Name: Virtual Disk 0
```

3.6 Setting external storage enclosure asset tag by using Redfish 2016

The Redfish 2016 storage APIs also enable the detailed inventory and management of external storage enclosures. For example, the asset tag for an external storage enclosure can be set by using a PATCH command on the URI “redfish/v1/Chassis/{external enclosure instance ID}/Settings”.

This example from the “SetEnclosureAssetTagRedfish.py” script in the Dell EMC Redfish Scripting GitHub shows the URI, payload, and PATCH command to set an enclosure’s asset tag:

```
url = 'https://%s/redfish/v1/Chassis/%s/Settings' % (idrac_ip,
external_enclosure)
    payload={"AssetTag":asset_tag}
    headers = {'content-type': 'application/json'}
    response = requests.patch(url, data=json.dumps(payload), headers=headers,
verify=False,auth=(idrac_username,idrac_password))
```

This example uses the “SetEnclosureAssetTagRedfish.py” script to set a storage enclosure asset tag:

1. View the storage controllers detected for the target server

```
C:\Python27>SetEnclosureAssetTagRedfish.py -ip 192.168.0.130 -u root -p calvin -
c y
- Server controller(s) detected -
RAID.Slot.2-1
PCIeExtender.Slot.1
AHCI.Embedded.1-1
AHCI.Embedded.2-1
PCIeExtender.Slot.7
PCIeExtender.Slot.
```

2. Get the external storage enclosures attached to the storage controller adapter RAID.Slot.2-1:

```
C:\Python27>SetEnclosureAssetTagRedfish.py -ip 192.168.0.130 -u root -p calvin -
e RAID.Slot.2-1
- Supported External Enclosures to Set Asset Tag for controller RAID.Slot.2-1 -
Enclosure.External.0-2:RAID.Slot.2-1
Enclosure.External.1-3:RAID.Slot.2-1
Enclosure.External.1-2:RAID.Slot.2-1
Enclosure.External.0-3:RAID.Slot.2-1
Enclosure.External.0-1:RAID.Slot.2-1
Enclosure.External.1-1:RAID.Slot.2-1
Enclosure.External.1-0:RAID.Slot.2-1
Enclosure.External.0-0:RAID.Slot.2-1
```

3. Before setting the storage enclosure’s asset tag for “Enclosure.External.1-2:RAID.Slot.2-1”, check the current asset tag value.

```
C:\Python27>SetEnclosureAssetTagRedfish.py -ip 192.168.0.130 -u root -p calvin -
-asset Enclosure.External.0-2:RAID.Slot.2-1
- WARNING, current asset tag for Enclosure.External.0-2:RAID.Slot.2-1 is: 645GVT
```

4. Set the asset tag to a new value “ABC123”, which implies that the server does not require a reboot to apply the configuration changes.

```
C:\Python27>SetEnclosureAssetTagRedfish.py -ip 192.168.0.130 -u root -p calvin -
x Enclosure.External.0-2:RAID.Slot.2-1 -a ABC123 -j r
- PASS: PATCH command passed to set asset tag pending value to "ABC123", status
code 200 returned
- PASS: PATCH command passed to create real time config job, status code 202
returned
- PASS, JID_270868118910 real time jid successfully created

- WARNING, JobStatus not completed, current status is: "New", precent completion
is: "0"
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "50"
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "50"
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "50"
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "99"
--- PASS, Final Detailed Job Status Results ---

JobState: Completed
Description: Job Instance
CompletionTime: 2018-05-23T16:11:41
PercentComplete: 100
StartTime: TIME_NOW
MessageId: PR19
Message: Job completed successfully.
EndTime: TIME_NA
Id: JID_270868118910
JobType: RealTimeNoRebootConfiguration
Name: Config:RAID:RAID.Slot.2-1
```

5. Finally, ensure that the asset tag has been updated to “ABC123”.

```
C:\Python27>SetEnclosureAssetTagRedfish.py -ip 192.168.0.130 -u root -p calvin -
-asset Enclosure.External.0-2:RAID.Slot.2-1
- WARNING, current asset tag for Enclosure.External.0-2:RAID.Slot.2-1 is: ABC123
```

3.7 Cryptographically erase drives by using Redfish 2016

The Redfish 2016 storage APIs support the cryptographic erase of ISE, SED, and PCIe SSD (NVMe) drives. Supported devices can be cryptographically erased by using a POST command with the target drive URI “redfish/v1/Systems/System.Embedded.1/Storage/Drives/{secure drive instance ID}/Actions/Drive.SecureErase”.

Note: Non-ISE hard drives (HDDs) cannot be erased by using this feature. If you want to erase non-ISE HDDs on the 14th generation of PowerEdge servers, you can use the System Erase function available in the

Dell Lifecycle Controller, which supports erasure of all types of server storage. For details about using System Erase, see the whitepaper Securing 14th generation Dell EMC PowerEdge servers with System Erase.

Note: The drive selected for erase by using Redfish 2016 must not be a member of a VD. Selecting such a drive for erase will return an error. Drives must be removed from a VD prior to attempting Redfish 2016 erase.

Here is an example from the "SecureEraseRedfish.py" script in the Dell EMC Redfish Scripting GitHub repository of URI and POST command to erase a drive:

```
url =
'https://%s/redfish/v1/Systems/System.Embedded.1/Storage/Drives/%s/Actions/Drive
.SecureErase' % (idrac_ip, secure_erase_device)
    headers = {'content-type': 'application/json'}
    response = requests.post(url, headers=headers,
verify=False,auth=(idrac_username,idrac_password))
```

The example below uses the "SecureEraseDevicesRedfish.py" script to erase an ISE drive.

1. Ensure that drives are "detected controller RAID.Slot.6-1".

```
C:\Python27>SecureEraseDevicesRedfish.py -ip 192.168.0.130 -u root -p calvin -d
RAID.Slot.6-1
- Drive(s) detected for RAID.Slot.6-1 -

Disk.Bay.0:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.1:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.2:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.3:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.4:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.5:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.6:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.7:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.8:Enclosure.Internal.0-1:RAID.Slot.6-1
Disk.Bay.9:Enclosure.Internal.0-1:RAID.Slot.6-1
```

2. Select the drive instance ID of the drive to be erased and pass to script.

```
C:\Python27>SecureEraseDevicesRedfish.py -ip 192.168.0.130 -u root -p calvin -s
Disk.Bay.0:Enclosure.Internal.0-1:RAID.Slot.6-1
- PASS: POST command passed to secure erase drive
"Disk.Bay.0:Enclosure.Internal.0-1:RAID.Slot.6-1", status code 202 returned
- PASS, "realtime" JID_270977416565 jid successfully created for secure erase
drive "Disk.Bay.0:Enclosure.Internal.0-1:RAID.Slot.6-1"

- WARNING, JobStatus not completed, current status is: "New", precent completion
is: "0"
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "1"
<Output edited for brevity>
- WARNING, JobStatus not completed, current status is: "Job in progress.",
precent completion is: "99"
--- PASS, Final Detailed Job Status Results ---
```

```
JobState: Completed
Description: Job Instance
CompletionTime: 2018-05-23T12:50:16
PercentComplete: 100
StartTime: TIME_NOW
MessageId: PR19
Message: Job completed successfully.
EndTime: TIME_NA
Id: JID_270977416565
JobType: RealTimeNoRebootConfiguration
Name: Config:RAID:RAID.Slot.6-1
```

3.8 Limitations when using Redfish 2016 standard storage APIs for storage configuration

There are several limitations on job workflows when using Redfish 2016 standard APIs for storage configurations:

- Stacking multiple storage configuration operations and running them within a single iDRAC job is not supported. For example, a certain storage management task might require the deletion of a virtual disk followed immediately by the creation of a new VD. While the user may want to perform both steps in a single iDRAC job, this is not possible under the current Redfish standard. These steps must be performed by separate steps, using one job to delete the virtual disk and a second job to create the new virtual disk with each job being monitored for completion before initiating another job
- Creating multiple sliced RAID volumes within a single job is also not supported. For example, a storage management task might require the creation of sixteen 2G RAID 0 virtual disks; this is not possible using a single iDRAC job within the current Redfish standard. The current storage APIs require that only a single sliced RAID virtual disk be created within an iDRAC job, the job be monitored for completion and the balance of virtual disks be created by repeating these steps.

3.9 Benefits of using the Dell EMC Server Configuration Profile feature

For certain storage management tasks, the Dell EMC Server Configuration Profile (SCP) feature can greatly simplify operations:

- Multiple storage management operations can be stacked within a single SCP import operation. For example, an SCP file, imported via a single iDRAC RESTful API call, can reset a PERC controller, create multiple virtual disks, initialize the virtual disks, set an encryption key, and lock the virtual disks, all performed via a single iDRAC job
- A range of PERC controller features are supported using an SCP which are not currently supported by Redfish 2016 standard storage APIs. These features include:
    - Converting disks between non-RAID / RAID state
    - Changing or removing a controller encryption key
    - Locking a RAID volume
    - Assigning / Un-assigning global or dedicated hot spares

- Creating a virtual disk with RAID levels 6 and 60
- Advanced span levels
- Resetting the PERC controller
- Clearing and importing foreign configurations
- Setting controller / virtual disk properties such as Background Initialization and Consistency Check rates
- Creating multiple RAID volumes

For more information about using SCP features for storage configuration, see the following technical white papers available on Dell Techcenter: RESTful Server Configuration with iDRAC REST API and Server cloning by using Server Configuration Profiles (SCP) on Dell EMC PowerEdge servers.

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 storage functions including detailed inventory, monitoring and configuration of storage controllers, enclosures and physical and virtual drives.

Dell EMC is a committed leader in the development and implementation of open, industry standards. Supporting Redfish within the 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