

# Automating Dell EMC PowerEdge Server Management by using iDRAC REST API with DMTF Redfish and Microsoft PowerShell

Texas Roemer, Test Principal Engineer  
Paul Rubin, Sr. Product Manager

Dell EMC Server Solutions

August 2017

## Revisions

Date	Description
August 2017	Initial release

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.

Copyright © 2017 Dell Inc. or its subsidiaries. All Rights Reserved. Dell, EMC, and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be the property of their respective owners. Published in the USA [8/24/2017]

Dell believes the information in this document is accurate as of its publication date. The information is subject to change without notice.

# Contents

Revisions.....	2
Executive summary.....	4
Introduction .....	4
The Redfish management standard .....	5
Microsoft PowerShell .....	6
1 iDRAC REST API with Redfish .....	7
2 Microsoft PowerShell prerequisites .....	8
3 Operating the iDRAC REST API with Redfish by using PowerShell.....	9
3.0 Setting power control properties (Set-PowerControlREDFISH) .....	10
3.1 Configuring BIOS setting (Set-OneBiosAttributeREDFISH).....	11
3.2 Setting an iDRAC password (Set-IdracUserPasswordREDFISH).....	12
3.3 Updating server firmware (Set-UpdateOneDeviceREDFISH) .....	13
3.4 Exporting or importing Server Configuration Profile (Set-ExportImportServerConfigurationProfileREDFISH).....	14
3.5 Getting iDRAC Lifecycle logs (Get-IdracLifecycleLogsREDFISH).....	17
4 Detailing cmdlet code for iDRAC REST API with Redfish .....	18
Conclusion .....	22
References.....	22

## Executive summary

The growing scale of cloud- and web-based data center infrastructure is reshaping the requirements of IT administrators worldwide. 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 requirement 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.

Microsoft PowerShell is an automation platform and scripting language for Microsoft Windows and Windows Server that simplifies the management of systems. Unlike other text-based shell scripts, Microsoft PowerShell harnesses the power of the Microsoft .NET Framework, providing rich objects and a massive set of built-in functionality for taking control of a Windows environments.

This technical white paper provides an overview of using Microsoft PowerShell to script operations of the integrated Dell Remote Access Controller (iDRAC) with Lifecycle Controller REST API with Redfish. This technical white paper provides tips for getting started with Microsoft PowerShell scripting for iDRAC REST and Redfish and describes the key use cases.

## Introduction

Since the inception of the x86 server in the late 1980s, IT administrators have sought the methods 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 due to 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 where the redundancy is provided in the software layer, making scalable management interfaces more critical.

To meet such market requirements, a new, unifying management standard was required. That standard— DMTF Redfish— is a next-generation management standard that uses a data model representation inside a hypermedia RESTful interface. Dell EMC support Redfish within the iDRAC with Lifecycle Controller REST API on 12<sup>th</sup>, 13<sup>th</sup>, and 14<sup>th</sup> generation Dell EMC PowerEdge servers. Automating server management operations by using iDRAC REST with Redfish is readily performed by using Microsoft PowerShell, a powerful automation platform and scripting language.

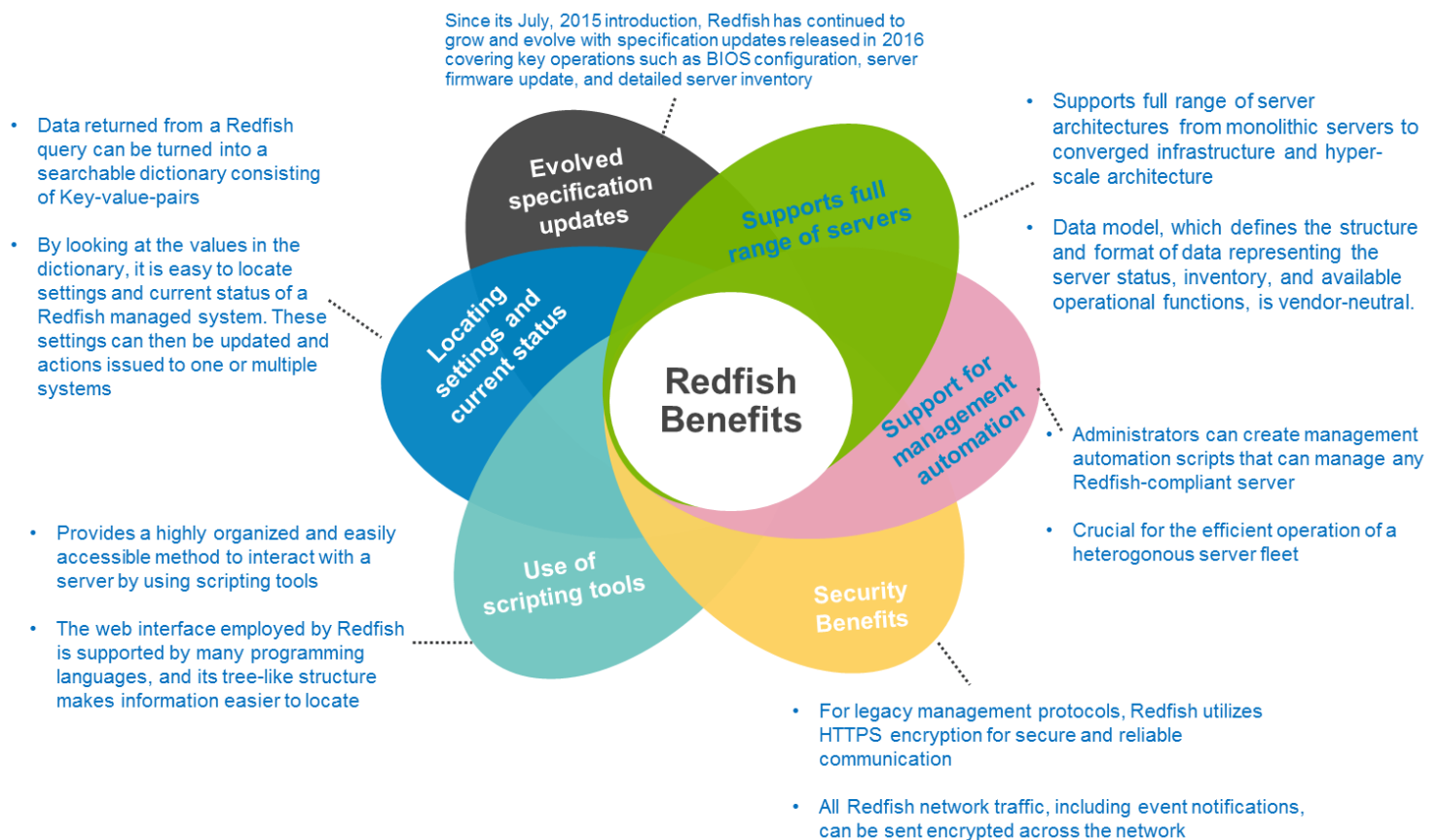
# The Redfish management standard

There are various Out-of-Band (OOB) Systems Management standards available in the industry. 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 requires, 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. Its key benefits include:

- Increased simplicity and usability
- Encrypted connections and generally heightened security
- A programmatic interface that can easily be controlled through scripts
- Ability to meet the Open Compute Project's Remote Machine Management requirements
- Based on widely-used standards for web APIs and data formats

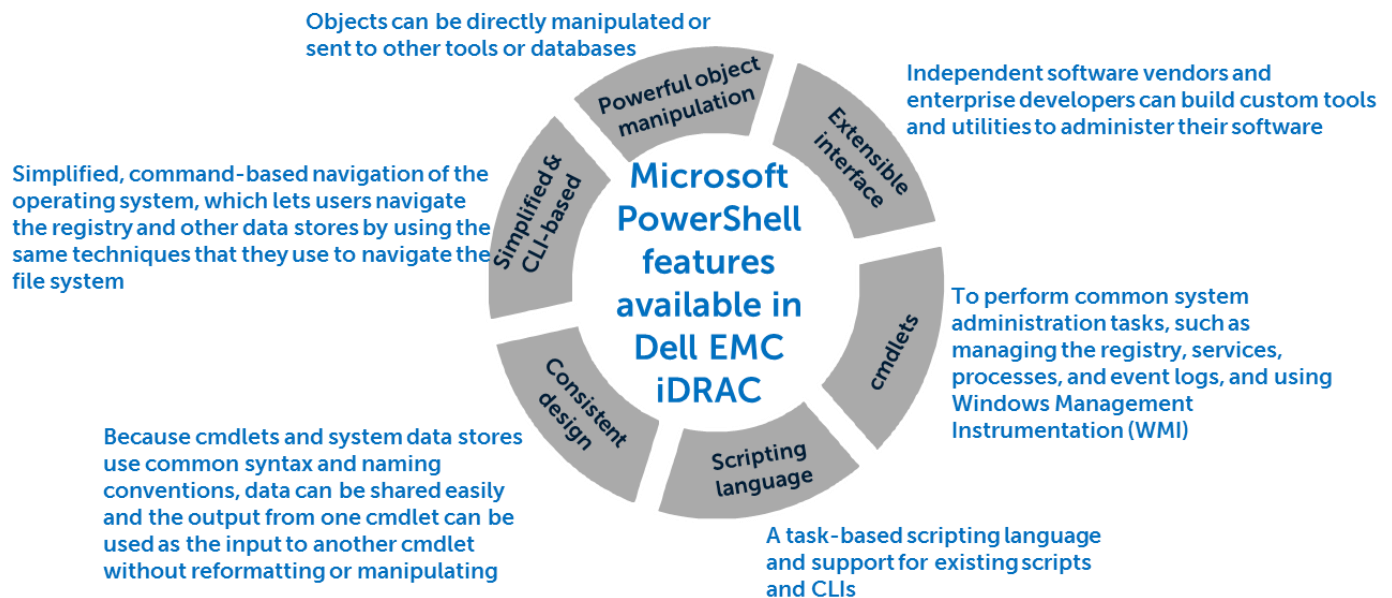


Redfish is supported by iDRAC with Lifecycle Controller REST API in the 12<sup>th</sup>, 13<sup>th</sup>, and 14<sup>th</sup> generation PowerEdge servers. For an overview of Redfish and iDRAC REST API, see the white paper [Implementation of the DMTF Redfish API on Dell EMC PowerEdge Servers](#) available on the Dell Techcenter website.

## Microsoft PowerShell

Microsoft PowerShell is a task-based CLI shell and scripting language designed for system administration automation. PowerShell helps IT professionals and power users to control and automate the administration of the Windows operating system and applications that run on Windows.

PowerShell commands, called cmdlets, enable management of computers from the CLI. PowerShell providers can access data stores, such as the Windows registry and certificate store, as easily as accessing the server file system. PowerShell also has a rich expression parser and a fully developed scripting language.



While primarily a Windows OS-oriented tool, PowerShell is also available in open source form, capable of operating in a variety of operating environments including Red Hat Enterprise Linux (RHEL), CentOS, Debian, Ubuntu, OpenSuSe, and Docker. For more information, see the PowerShell GitHub repository: <https://github.com/PowerShell/PowerShell>.

# 1 iDRAC REST API with Redfish

To support the DMTF Redfish standard, the iDRAC with Lifecycle Controller supports a RESTful API in addition to support for the IPMI, SNMP, and WS-Man standard APIs. The iDRAC REST API builds on the Redfish standard to provide a RESTful interface for the Dell EMC value-add operations including:

- Information about all the iDRAC with Lifecycle Controller out-of-band services—web server, SNMP, virtual media, SSH, Telnet, IPMI, and KVM.
- Expanded storage subsystem reporting that covers controllers, enclosures, and drives.
- For the PowerEdge FX2 modular, detailed chassis information that covers power supply units (PSUs), temperature, and fans.
- With the iDRAC Service Module (iSM) installed under the server OS, the API provides detailed inventory and status reporting for host network interfaces information such as IP address, subnet mask, and gateway for the Host OS.

## 2 Microsoft PowerShell prerequisites

Before running any Microsoft PowerShell cmdlets by using the iDRAC REST API with Redfish, the following prerequisites must be fulfilled:

- Install PowerShell 5.0 or later. Run `Get-Host` to verify the PowerShell version.
- Set `ExecutionPolicy` to “RemoteSigned”. To check the current setting, run `Get-ExecutionPolicy`. To change the setting, run `Set-ExecutionPolicy`.
- Ensure that Redfish is enabled on the iDRAC. By default, the Redfish service is enabled on iDRAC.
- If a cmdlet requires to be automatically imported when starting a PowerShell session, create a directory with the same name as the cmdlet in the directory path `C:\Users\'your user name'\Documents\WindowsPowerShell\Modules`. For example, goal is for the `Set-PowerControlRedfish.psm1` cmdlet to automatically be imported when a PowerShell session is started. Create the directory `C:\Users\'user name'\Documents\WindowsPowerShell\Modules\Set-PowerControlREDFISH` and copy the cmdlet into this directory path.

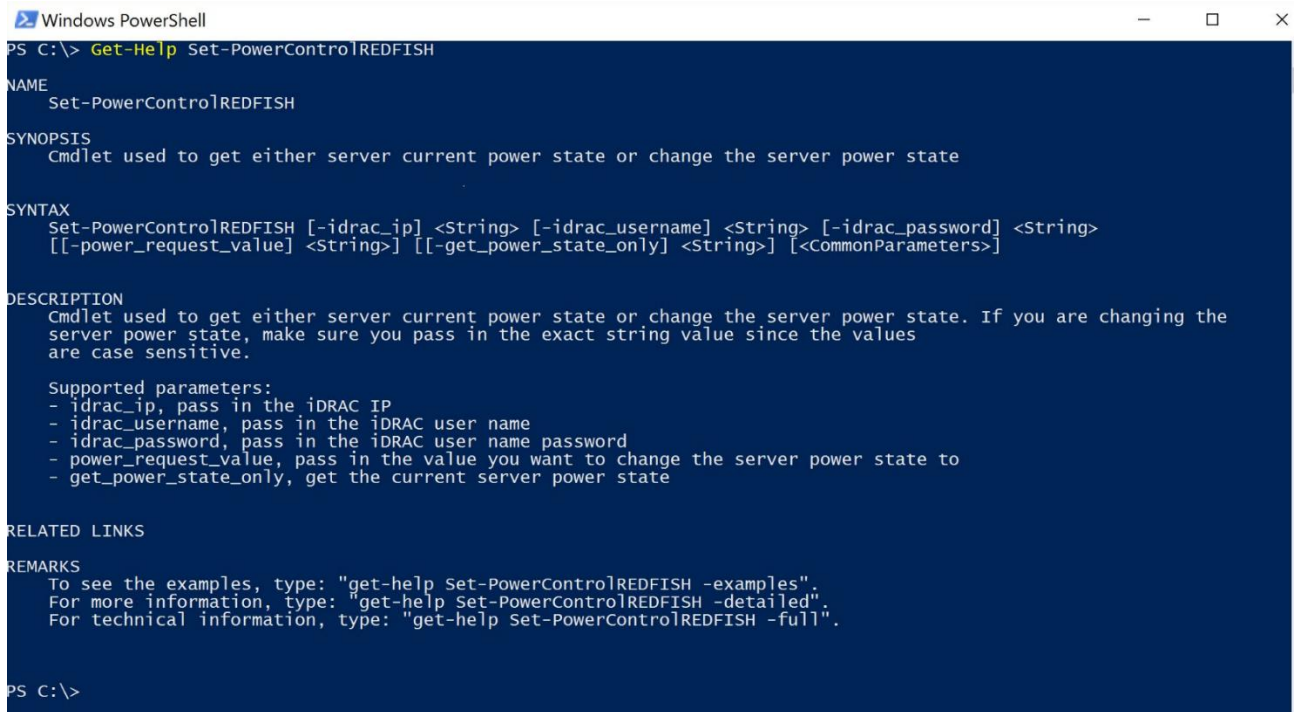


### 3 Operating the iDRAC REST API with Redfish by using PowerShell

The following PowerShell scripts implement key use cases for the iDRAC REST API with Redfish including server power control, BIOS configuration, firmware update, RESTful server configuration, firmware update, and log export. The scripts have been tested on a PowerEdge R740 server with iDRAC9 firmware version 3.00.00.00. The scripts used in this white paper are available from the GitHub open source repository <https://github.com/dell/iDRAC-Redfish-Scripting>.

**Note:** It is recommended to run `Get-Help` on a cmdlet before executing it. This will return helpful information explaining the functionality of the cmdlet and detailing the required or optional parameters. Run `Get-Help 'cmdlet name' -examples` which will return examples of executing the cmdlet.

Sample screen shots show running the `Get-Help` on the cmdlet `Set-PowerControlRedfish`:



```
Windows PowerShell
PS C:\> Get-Help Set-PowerControlRedfish

NAME
    Set-PowerControlRedfish

SYNOPSIS
    Cmdlet used to get either server current power state or change the server power state

SYNTAX
    Set-PowerControlRedfish [-idrac_ip] <String> [-idrac_username] <String> [-idrac_password] <String>
    [[-power_request_value] <String>] [[-get_power_state_only] <String>] [<CommonParameters>]

DESCRIPTION
    Cmdlet used to get either server current power state or change the server power state. If you are changing the
    server power state, make sure you pass in the exact string value since the values
    are case sensitive.

    Supported parameters:
    - idrac_ip, pass in the iDRAC IP
    - idrac_username, pass in the iDRAC user name
    - idrac_password, pass in the iDRAC user name password
    - power_request_value, pass in the value you want to change the server power state to
    - get_power_state_only, get the current server power state

RELATED LINKS

REMARKS
    To see the examples, type: "get-help Set-PowerControlRedfish -examples".
    For more information, type: "get-help Set-PowerControlRedfish -detailed".
    For technical information, type: "get-help Set-PowerControlRedfish -full".

PS C:\>
```

Figure 1 Running Get-Help on the Set-PowerControlRedfish cmdlet

```
Windows PowerShell
PS C:\> Get-Help Set-PowerControlRedfish -examples

NAME
    Set-PowerControlRedfish

SYNOPSIS
    Cmdlet used to get either server current power state or change the server power state

----- EXAMPLE 1 -----

PS C:\>Set-PowerControlRedfish -idrac_ip 192.168.0.120 -idrac_username root -idrac_password calvin
-get_power_state_only y

# Example to get current server power state

----- EXAMPLE 2 -----

PS C:\>Set-PowerControlRedfish -idrac_ip 192.168.0.120 -idrac_username root -idrac_password calvin
-power_request_value On

# Example to power ON the server

PS C:\>
```

Figure 2 Get-Help with examples on the Set-PowerControlRedfish cmdlet

### 3.0 Setting power control properties (Set-PowerControlRedfish)

The Set-PowerControlRedfish PowerShell cmdlet can obtain the server's current power state, view the possible power control options, and change the server power state. The sample screen shots show the process of getting the current server power state and then performing a graceful restart of the server:

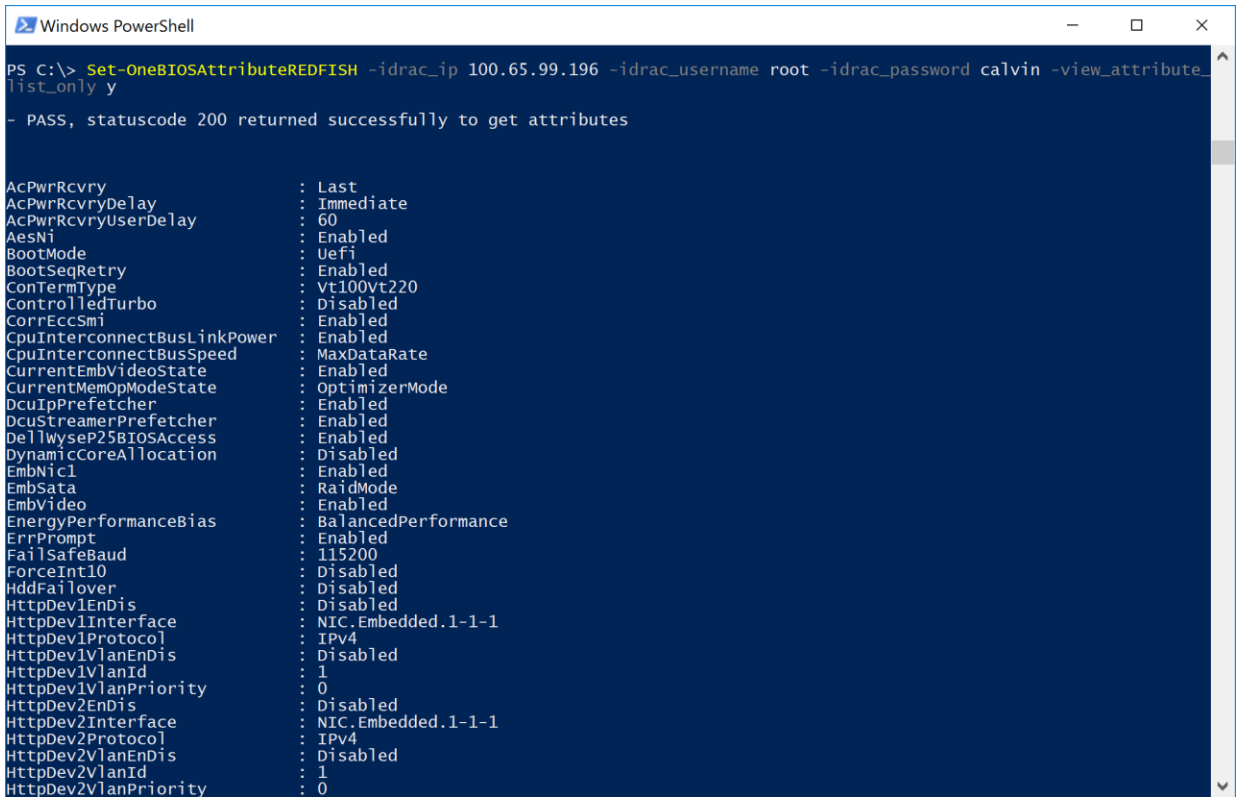
```
Windows PowerShell
PS C:\> Set-PowerControlRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin -get_power_state_only y
y
- PASS, statuscode 200 returned successfully to get current power state
- WARNING, Server current power state is ON
Supported power control values are:
- On
- ForceOff
- GracefulRestart
- GracefulShutdown
PS C:\> Set-PowerControlRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin -power_request_value
GracefulRestart
- PASS, statuscode 204 returned, power control operation success for: GracefulRestart
PS C:\>
```

Figure 3 Viewing server power state by running the Set-PowerControlRedfish cmdlet

### 3.1 Configuring BIOS setting (Set-OneBiosAttributeREDFISH)

The `Set-OneBiosAttributeRedfish` cmdlet is used to get the current value of BIOS attributes or to set a single BIOS attribute to a new value:

1. Run cmdlet to get all the BIOS attributes and their current values:

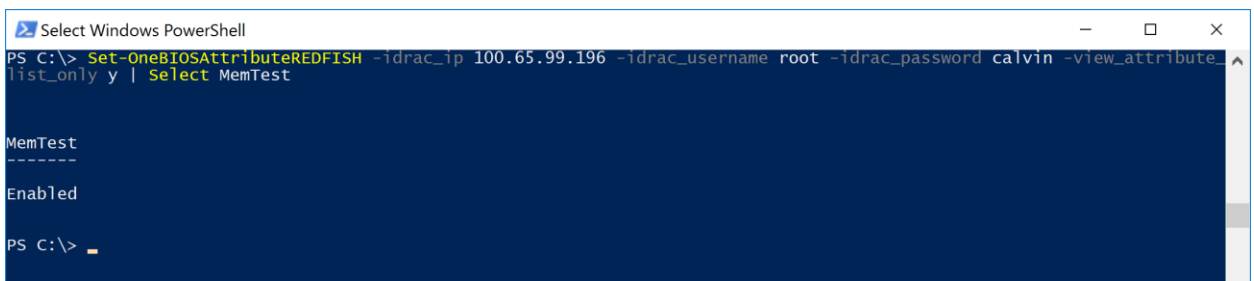


```
Windows PowerShell
PS C:\> Set-OneBiosAttributeRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin -view_attribute_
list_only y
- PASS, statuscode 200 returned successfully to get attributes

ACpwrRcvry           : Last
ACpwrRcvryDelay      : Immediate
ACpwrRcvryUserDelay  : 60
AesNi                : Enabled
BootMode             : Uefi
BootSeqRetry         : Enabled
ConTermType          : Vt100Vt220
ControlledTurbo      : Disabled
CorrEccSmi           : Enabled
CpuInterconnectBusLinkPower : Enabled
CpuInterconnectBusSpeed : MaxDataRate
CurrentEmbVideoState : Enabled
CurrentMemOpModeState : OptimizerMode
DcuIpPrefetcher      : Enabled
DcuStreamerPrefetcher : Enabled
DeLlyseP25BIOSAccess : Enabled
DynamicCoreAllocation : Disabled
EmbN1c1              : Enabled
EmbSata               : RaidMode
EmbVideo             : Enabled
EnergyPerformanceBias : BalancedPerformance
ErrPrompt            : Enabled
FailSafeBaud         : 115200
ForceInt10           : Disabled
HddFailover          : Disabled
HttpDev1EnDis        : Disabled
HttpDev1Interface     : NIC.Embedded.1-1-1
HttpDev1Protocol      : IPv4
HttpDev1VlanEnDis    : Disabled
HttpDev1VlanId        : 1
HttpDev1VlanPriority  : 0
HttpDev2EnDis        : Disabled
HttpDev2Interface     : NIC.Embedded.1-1-1
HttpDev2Protocol      : IPv4
HttpDev2VlanEnDis    : Disabled
HttpDev2VlanId        : 1
HttpDev2VlanPriority  : 0
```

Figure 4 Viewing BIOS attribute values with Set-OneBiosAttributeRedfish cmdlet

2. To obtain the value of a single attribute, enter the vertical symbol (|) and use the **Select** built-in command to access the desired attribute. Here is an example for using **Select** to view the **MemTest** attribute:



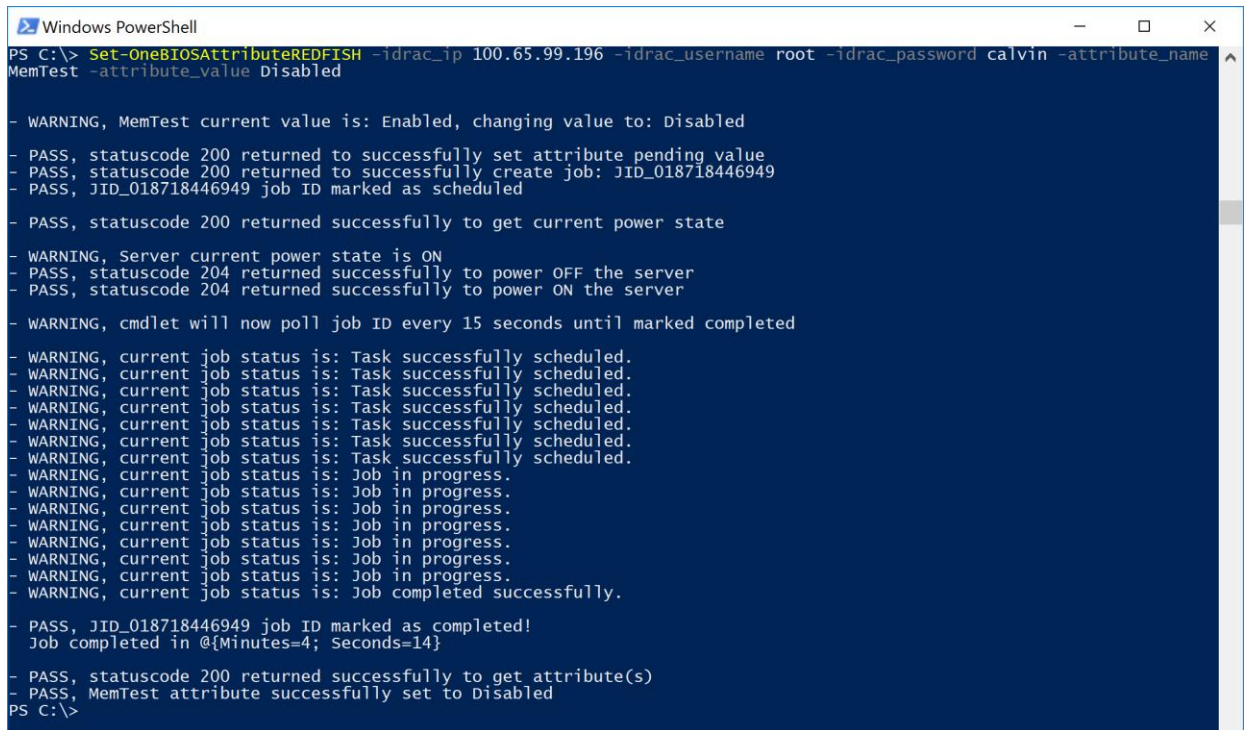
```
Select Windows PowerShell
PS C:\> Set-OneBiosAttributeRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin -view_attribute_
list_only y | Select MemTest

MemTest
-----

Enabled

PS C:\>
```

### 3. Run `Set-OneBIOSAttributeRedfish` to set the `MemTest` attribute to `Disabled`:



```
Windows PowerShell
PS C:\> Set-OneBIOSAttributeRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin -attribute_name MemTest -attribute_value Disabled

- WARNING, MemTest current value is: Enabled, changing value to: Disabled
- PASS, statuscode 200 returned to successfully set attribute pending value
- PASS, statuscode 200 returned to successfully create job: JID_018718446949
- PASS, JID_018718446949 job ID marked as scheduled

- PASS, statuscode 200 returned successfully to get current power state
- WARNING, Server current power state is ON
- PASS, statuscode 204 returned successfully to power OFF the server
- PASS, statuscode 204 returned successfully to power ON the server

- WARNING, cmdlet will now poll job ID every 15 seconds until marked completed

- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job completed successfully.

- PASS, JID_018718446949 job ID marked as completed!
  Job completed in @{Minutes=4; Seconds=14}

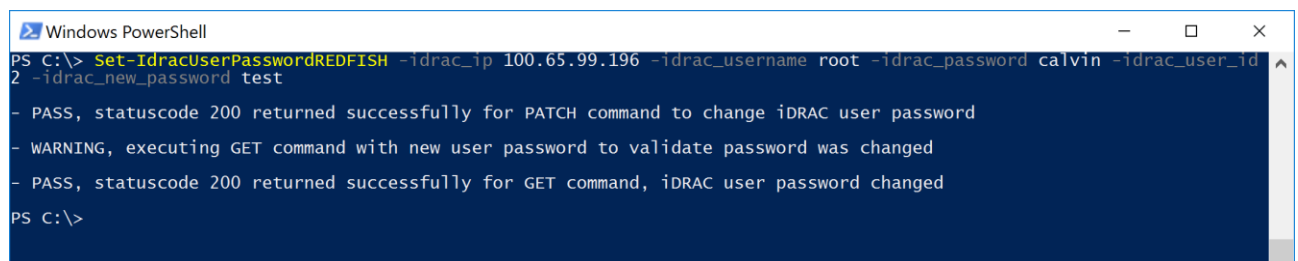
- PASS, statuscode 200 returned successfully to get attribute(s)
- PASS, MemTest attribute successfully set to Disabled
PS C:\>
```

Figure 5 Setting a single BIOS attribute with `Set-OneBIOSAttributeRedfish`

## 3.2 Setting an iDRAC password (`Set-IdracUserPasswordRedfish`)

The `Set-IdracUserPasswordRedfish` cmdlet changes an iDRAC user's password and then runs a `GET` command by using the new password to verify that the user password was changed. To run the cmdlet, the iDRAC user ID index number is required with the current user name and password. If required, run `GET` on `redfish/v1/Managers/iDRAC.Embedded.1/Accounts/` to determine the iDRAC User ID index number for the user whose password is to be changed.

In the following example, the root user password is changed from `calvin` to `test`. The user ID index for the "root" user is "2".



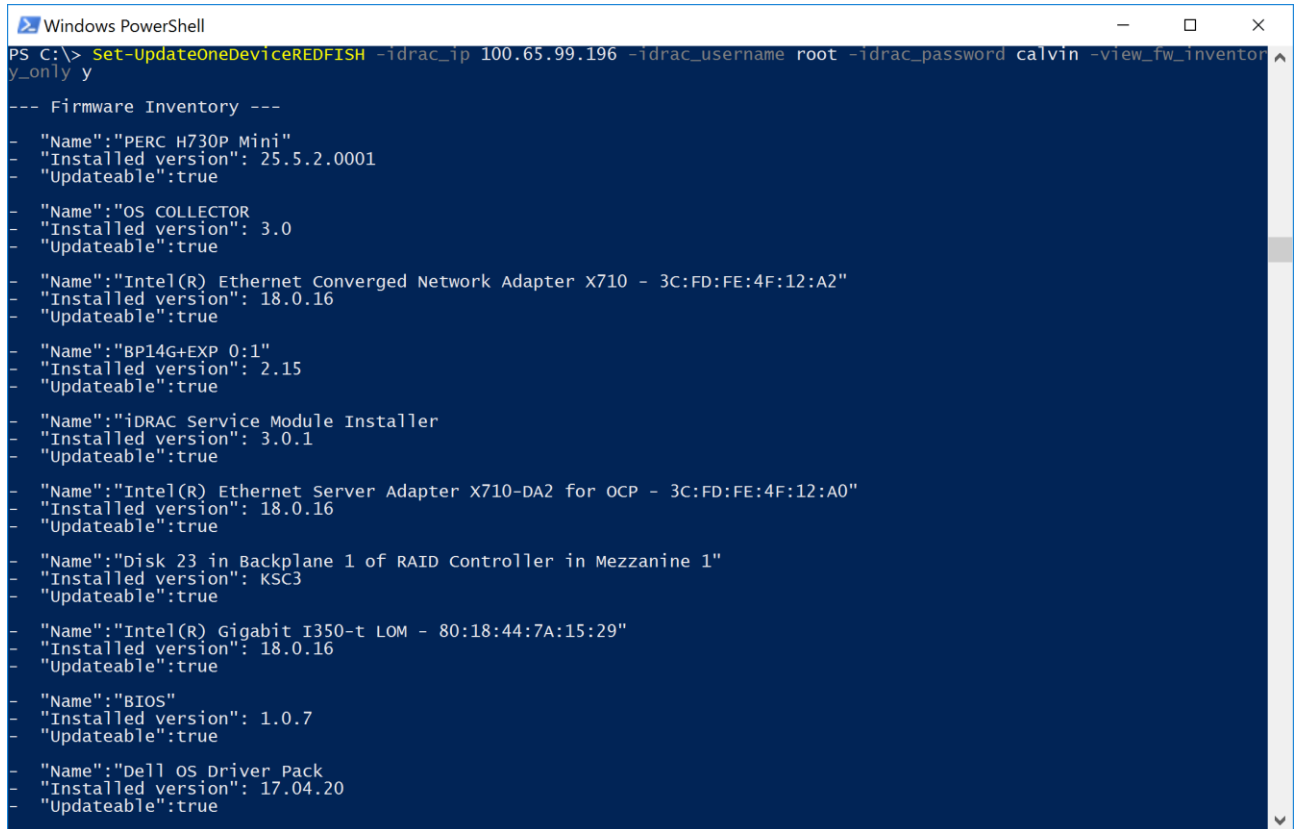
```
Windows PowerShell
PS C:\> Set-IdracUserPasswordRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin -idrac_user_id 2 -idrac_new_password test

- PASS, statuscode 200 returned successfully for PATCH command to change iDRAC user password
- WARNING, executing GET command with new user password to validate password was changed
- PASS, statuscode 200 returned successfully for GET command, iDRAC user password changed
PS C:\>
```

Figure 6 Changing iDRAC user password with `set-IdracUserPasswordRedfish` cmdlet

### 3.3 Updating server firmware (Set-UpdateOneDeviceRedfish)

The `Set-UpdateOneDeviceRedfish` cmdlet provides an inventory of the current firmware versions for devices in the server which iDRAC supports for updates. Also supports performing a firmware update on a single device. The firmware update file must be stored on the same system from which the cmdlet is run; the update file content is streamed to the iDRAC within the API. The following example illustrates obtaining the current firmware inventory:



```
Windows PowerShell
PS C:\> Set-UpdateOneDeviceRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin -view_fw_inventory_only y

--- Firmware Inventory ---
- "Name": "PERC H730P Mini"
- "Installed version": 25.5.2.0001
- "Updateable": true

- "Name": "OS COLLECTOR"
- "Installed version": 3.0
- "Updateable": true

- "Name": "Intel(R) Ethernet Converged Network Adapter X710 - 3C:FD:FE:4F:12:A2"
- "Installed version": 18.0.16
- "Updateable": true

- "Name": "BP14G+EXP 0:1"
- "Installed version": 2.15
- "Updateable": true

- "Name": "iDRAC Service Module Installer"
- "Installed version": 3.0.1
- "Updateable": true

- "Name": "Intel(R) Ethernet Server Adapter X710-DA2 for OCP - 3C:FD:FE:4F:12:A0"
- "Installed version": 18.0.16
- "Updateable": true

- "Name": "Disk 23 in Backplane 1 of RAID Controller in Mezzanine 1"
- "Installed version": KSC3
- "Updateable": true

- "Name": "Intel(R) Gigabit I350-t LOM - 80:18:44:7A:15:29"
- "Installed version": 18.0.16
- "Updateable": true

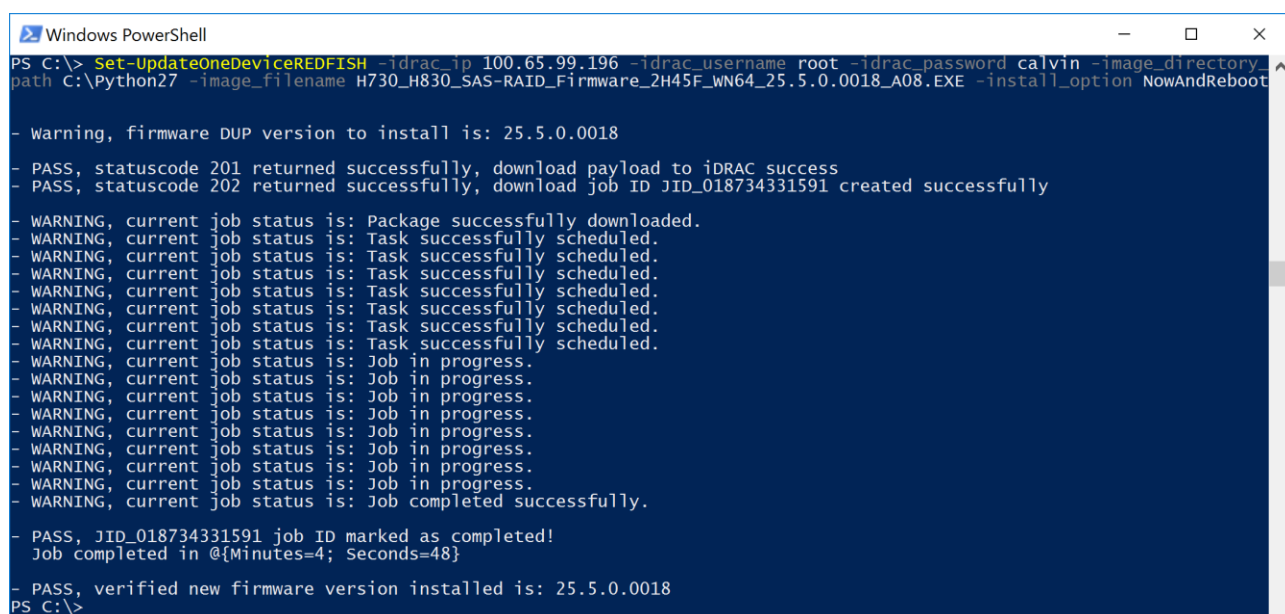
- "Name": "BIOS"
- "Installed version": 1.0.7
- "Updateable": true

- "Name": "Dell OS Driver Pack"
- "Installed version": 17.04.20
- "Updateable": true
```

Figure 7 Viewing server firmware inventory with Set-UpdateOneDeviceRedfish



In the screen shot, a PERC H730P Mini controller is updated from the firmware version 25.5.2.0001 to 25.5.0.0018. The update is applied with an immediate server restart. The Windows Update Package file for version 25.5.0.0018 is saved locally in the C:\Python27 directory before running the cmdlet.



```
Windows PowerShell
PS C:\> Set-UpdateOneDeviceRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin -image_directory
path C:\Python27 -image_filename H730_H830_SAS-RAID_Firmware_2H45F_WN64_25.5.0.0018_A08.EXE -install_option NowAndReboot

- Warning, firmware DUP version to install is: 25.5.0.0018
- PASS, statuscode 201 returned successfully, download payload to iDRAC success
- PASS, statuscode 202 returned successfully, download job ID JID_018734331591 created successfully
- WARNING, current job status is: Package successfully downloaded.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Task successfully scheduled.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job in progress.
- WARNING, current job status is: Job completed successfully.
- PASS, JID_018734331591 job ID marked as completed!
  Job completed in @{Minutes=4; Seconds=48}
- PASS, verified new firmware version installed is: 25.5.0.0018
PS C:\>
```

Figure 8 Updating PERC firmware using Set-UpdateOneDeviceRedfish cmdlet

### 3.4 Exporting or importing Server Configuration Profile (Set-ExportImportServerConfigurationProfileRedfish)

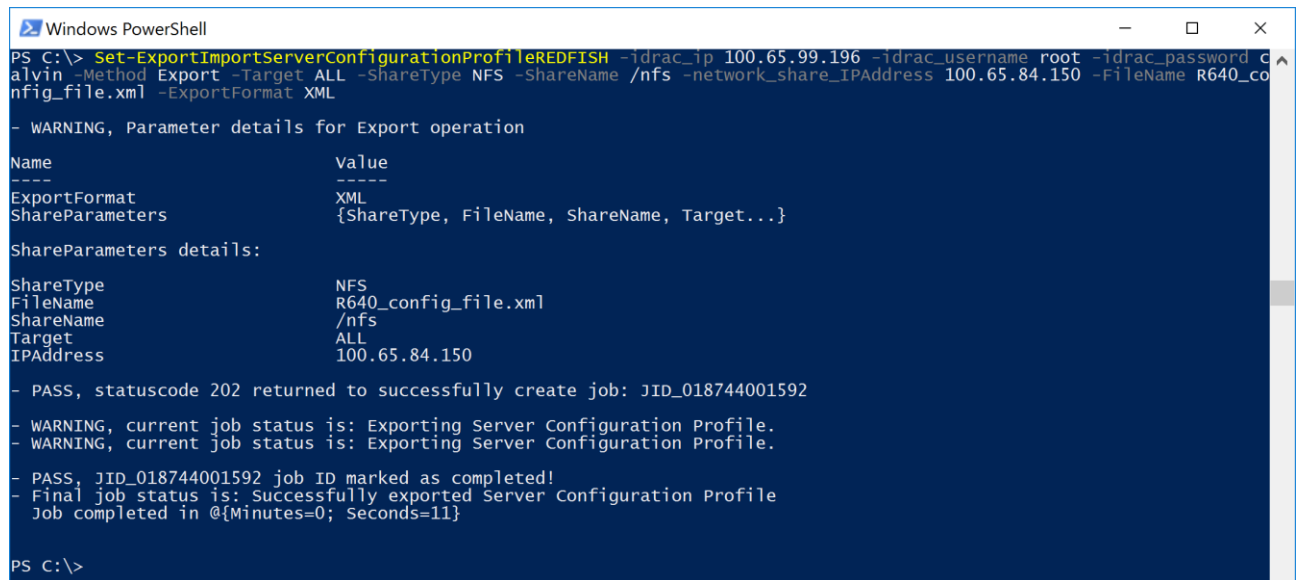
The Set-ExportImportServerConfigurationProfile cmdlet enables export and import of Server Configuration Profile (SCP) files from a network share. Beginning with 12th generation PowerEdge servers, iDRAC with Lifecycle Controller has featured the use of SCP files to configure BIOS, iDRAC, PERC RAID controller, and NIC or HBA settings in a single profile.

This greatly simplifies bare-metal deployments and scale-out operations by removing complexity from server configuration. Rather than manually interacting with the BIOS on the System Setup (F2) and company logo (F10) pages. Or, by writing complex scripts, administrators can set up an initial “gold” configuration server, capture the settings into an SCP file, modify the profile as required, and apply the profile across a pool of target servers.

In the example screen shot, an SCP is exported to an NFS, and the exported SCP is modified to make iDRAC attribute changes. After the changes are completed, the cmdlet command is used to import the SCP and apply the configuration changes to the iDRAC.

**Note:** Before running the SCP import, a recommended best practice is to run the import preview operation on the SCP file. By doing this, iDRAC validates the SCP, ensuring there are no issues in performing the import operation by using the SCP file.

The following screen shots describe the flow of importing preview before running the import:



```
PS C:\> Set-ExportImportServerConfigurationProfileRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin -Method Export -Target ALL -ShareType NFS -ShareName /nfs -network_share_IPAddress 100.65.84.150 -FileName R640_config_file.xml -ExportFormat XML

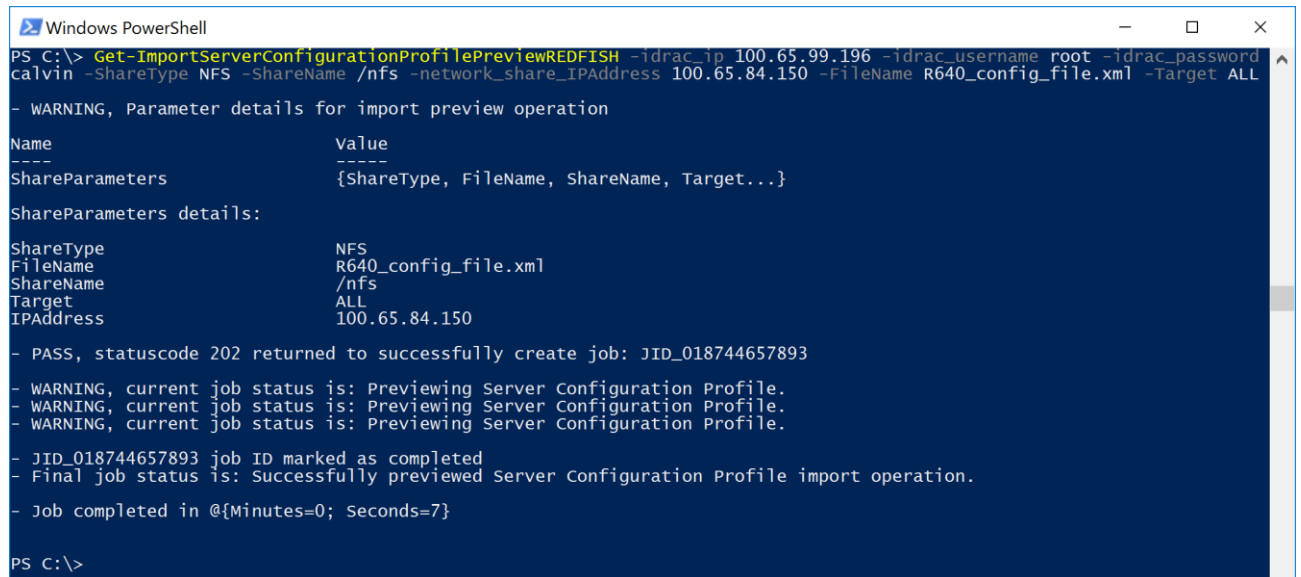
- WARNING, Parameter details for Export operation
Name                Value
----                -
ExportFormat        XML
ShareParameters     {ShareType, FileName, ShareName, Target...}

ShareParameters details:
ShareType           NFS
FileName            R640_config_file.xml
ShareName           /nfs
Target              ALL
IPAddress           100.65.84.150

- PASS, statuscode 202 returned to successfully create job: JID_018744001592
- WARNING, current job status is: Exporting Server Configuration Profile.
- WARNING, current job status is: Exporting Server Configuration Profile.
- PASS, JID_018744001592 job ID marked as completed!
- Final job status is: Successfully exported Server Configuration Profile
  Job completed in @{Minutes=0; Seconds=11}

PS C:\>
```

Figure 9 Exporting an SCP by using the Set-ExportImportServerConfigurationProfileRedfish cmdlet



```
PS C:\> Get-ImportServerConfigurationProfilePreviewRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin -ShareType NFS -ShareName /nfs -network_share_IPAddress 100.65.84.150 -FileName R640_config_file.xml -Target ALL

- WARNING, Parameter details for import preview operation
Name                Value
----                -
ShareParameters     {ShareType, FileName, ShareName, Target...}

ShareParameters details:
ShareType           NFS
FileName            R640_config_file.xml
ShareName           /nfs
Target              ALL
IPAddress           100.65.84.150

- PASS, statuscode 202 returned to successfully create job: JID_018744657893
- WARNING, current job status is: Previewing Server Configuration Profile.
- WARNING, current job status is: Previewing Server Configuration Profile.
- WARNING, current job status is: Previewing Server Configuration Profile.
- JID_018744657893 job ID marked as completed
- Final job status is: Successfully previewed Server Configuration Profile import operation.
- Job completed in @{Minutes=0; Seconds=7}

PS C:\>
```

Figure 10 Importing preview of SCP by using the Set-ExportImportServerConfigurationProfileRedfish cmdlet

```
Windows PowerShell
PS C:\> Set-ExportImportServerConfigurationProfileRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password c
alvin -Method Import -Target ALL -Sharetype NFS -ShareName /nfs -network_share_IPAddress 100.65.84.150 -FileName R640_co
nfig_file.xml

- WARNING, Parameter details for Import operation
Name                Value
----                -
ShareParameters     {ShareType, FileName, ShareName, Target...}
ShareParameters details:
ShareType           NFS
FileName            R640_config_file.xml
ShareName           /nfs
Target              ALL
IPAddress           100.65.84.150

- PASS, statuscode 202 returned to successfully create job: JID_018745111215
- WARNING, current job status is: Importing Server Configuration Profile.
- WARNING, current job status is: Importing Server Configuration Profile.
- PASS, JID_018745111215 job ID marked as completed!
- Final job status is: Successfully imported and applied Server Configuration Profile.
  Job completed in @{Minutes=0; Seconds=11}

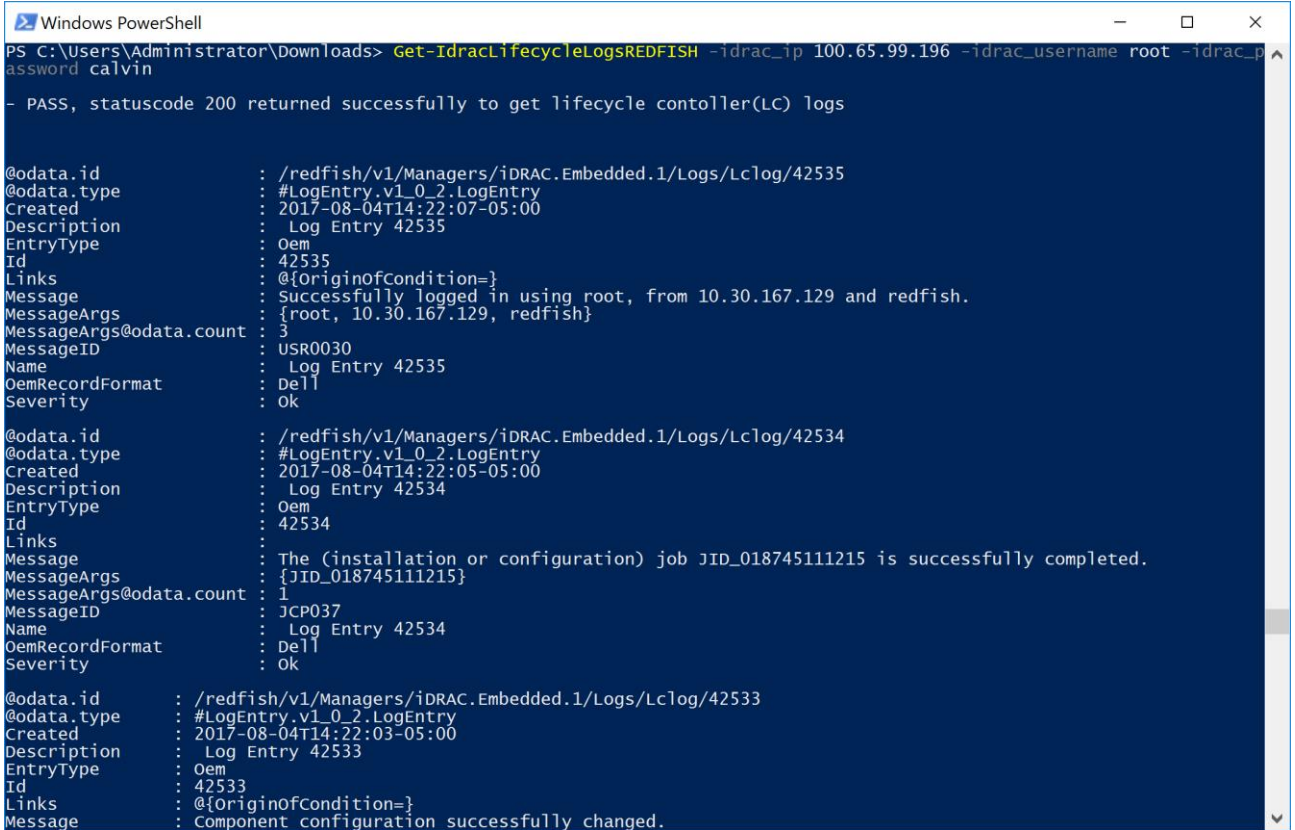
PS C:\>
```

Figure 11 Changing the iDRAC configuration by using an SCP import with the Set-ExportImportServerConfigurationProfileRedfish cmdlet



### 3.5 Getting iDRAC Lifecycle logs (Get-IdracLifecycleLogsREDFISH)

The `Get-IdracLifecycleLogsRedfish` cmdlet exports the Lifecycle Controller logs. The cmdlet outputs the logs to standard output but also captures the logs in a text file (`lifecycle_logs.txt`) with a timestamp indicating the time of the cmdlet execution.



```
Windows PowerShell
PS C:\Users\Administrator\Downloads> Get-IdracLifecycleLogsRedfish -idrac_ip 100.65.99.196 -idrac_username root -idrac_password calvin

- PASS, statuscode 200 returned successfully to get lifecycle controller(LC) logs

@odata.id      : /redfish/v1/Managers/iDRAC.Embedded.1/Logs/Lclog/42535
@odata.type    : #LogEntry.v1_0_2.LogEntry
Created        : 2017-08-04T14:22:07-05:00
Description    : Log Entry 42535
EntryType     : Oem
Id             : 42535
Links          : @{{OriginOfCondition=}}
Message       : Successfully logged in using root, from 10.30.167.129 and redfish.
MessageArgs    : {root, 10.30.167.129, redfish}
MessageArgs@odata.count : 3
MessageID     : USR0030
Name          : Log Entry 42535
OemRecordFormat : Dell
Severity      : Ok

@odata.id      : /redfish/v1/Managers/iDRAC.Embedded.1/Logs/Lclog/42534
@odata.type    : #LogEntry.v1_0_2.LogEntry
Created        : 2017-08-04T14:22:05-05:00
Description    : Log Entry 42534
EntryType     : Oem
Id             : 42534
Links          : 
Message       : The (installation or configuration) job JID_018745111215 is successfully completed.
MessageArgs    : {JID_018745111215}
MessageArgs@odata.count : 1
MessageID     : JCP037
Name          : Log Entry 42534
OemRecordFormat : Dell
Severity      : Ok

@odata.id      : /redfish/v1/Managers/iDRAC.Embedded.1/Logs/Lclog/42533
@odata.type    : #LogEntry.v1_0_2.LogEntry
Created        : 2017-08-04T14:22:03-05:00
Description    : Log Entry 42533
EntryType     : Oem
Id             : 42533
Links          : @{{OriginOfCondition=}}
Message       : Component configuration successfully changed.
```

Figure 12 Viewing and saving the Lifecycle Controller log with the `Get-IdracLifecycleLogsRedfish` cmdlet

## 4 Detailing cmdlet code for iDRAC REST API with Redfish

Here, the `Set-PowerControlRedfish` cmdlet is annotated with highlighted comments for each section of the code.

```
<#
# The first part of the cmdlet will always be the comment block explaining
description, examples, etc. which can get accessed when you run "Get-Help" on the
cmdlet.

_ author_ = Texas Roemer <Texas_Roemer@Dell.com>
_ version_ = 2.0

Copyright (c) 2017, Dell, Inc.

This software is licensed to you under the GNU General Public License,
version 2 (GPLv2). There is NO WARRANTY for this software, express or
implied, including the implied warranties of MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE. You should have received a copy of GPLv2
along with this software; if not, see
http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt
#>

<#
.Synopsis
    Cmdlet used to get either server current power state or change the server
power state
.DESCRPTION
    Cmdlet used to get either server current power state or change the server
power state. If you are changing the server power state, make sure you pass in
the exact string value since the values are case sensitive.

    Supported parameters:
    - idrac_ip, pass in the iDRAC IP
    - idrac_username, pass in the iDRAC user name
    - idrac_password, pass in the iDRAC user name password
    - power_request_value, pass in the value you want to change the server power
state to
    - get_power_state_only, get the current server power state
.EXAMPLE
    Set-PowerControlREDFISH -idrac_ip 192.168.0.120 -idrac_username root -idrac_
password calvin -get_power_state_only y
    # Example to get current server power state
.EXAMPLE
    Set-PowerControlREDFISH -idrac_ip 192.168.0.120 -idrac_username root -
idrac_password calvin -power_request_value On
    # Example to power ON the server
#>
```

```

function Set-PowerControlREDFISH {

# Within the function, define the supported parameters, indicating which are
mandatory and which are optional.
param(
    [Parameter(Mandatory=$True)]
    [string]$idrac_ip,
    [Parameter(Mandatory=$True)]
    [string]$idrac_username,
    [Parameter(Mandatory=$True)]
    [string]$idrac_password,
    [Parameter(Mandatory=$False)]
    [string]$power_request_value,
    [Parameter(Mandatory=$False)]
    [string]$get_power_state_only
)

# Function to ignore SSL certs. If you don't have a valid SSL certificate for the
iDRAC, pass in this function which will ignore SSL cert check.

function Ignore-SSLCertificates
{
    $Provider = New-Object Microsoft.CSharp.CSharpCodeProvider
    $Compiler = $Provider.CreateCompiler()
    $Params = New-Object System.CodeDom.Compiler.CompilerParameters
    $Params.GenerateExecutable = $false
    $Params.GenerateInMemory = $true
    $Params.IncludeDebugInformation = $false
    $Params.ReferencedAssemblies.Add("System.DLL") > $null
    $TASource=@'
        namespace Local.ToolkitExtensions.Net.CertificatePolicy
        {
            public class TrustAll : System.Net.ICertificatePolicy
            {
                public bool CheckValidationResult(System.Net.ServicePoint
sp,System.Security.Cryptography.X509Certificates.X509Certificate cert,
System.Net.WebRequest req, int problem)
                {
                    return true;
                }
            }
        }
    '@
    $TAResults=$Provider.CompileAssemblyFromSource($Params,$TASource)
    $TAAssembly=$TAResults.CompiledAssembly
    $TrustAll =
$TAAssembly.CreateInstance("Local.ToolkitExtensions.Net.CertificatePolicy.TrustAl
1")
    [System.Net.ServicePointManager]::CertificatePolicy = $TrustAll
}

Ignore-SSLCertificates

```

```

# Below code is needed to set up the credentials correctly for PowerShell to
process and accept them.
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::TLS12
$user = $idrac_username
$pass= $idrac_password
$secpasswd = ConvertTo-SecureString $pass -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential($user,
$secpasswd)

# Get current server power state

if ($get_power_state_only -eq "y")
{

# Define the Redfish URI which will be used for GET, PATCH or POST commands.
$u = https://$idrac_ip/redfish/v1/Systems/System.Embedded.1/
# Compile the GET Command
$result = Invoke-WebRequest -Uri $u -Credential $credential -Method Get -
UseBasicParsing
# Conditional statement to check if correct status code was returned by GET
command
if ($result.StatusCode -eq 200)
{
    Write-Host
    [String]::Format("- PASS, statuscode {0} returned successfully to get current
power state",$result.StatusCode)
}
else
{
    [String]::Format("- FAIL, statuscode {0} returned",$result.StatusCode)
    return
}
# Parse the output from the GET command with regex to get the current server
power state.
$get_content=$result.Content
$power_state=[regex]::Match($get_content, "PowerState.+?,").Captures[0].value
$power_state=$power_state -replace ("","")
$power_state=$power_state -split (":")

if ($power_state -eq '"On"')
{
Write-Host
Write-Host "- WARNING, Server current power state is ON"
}
else
{
Write-Host
Write-Host "- WARNING, Server current power state is OFF"
}
Write-Host
Write-Host "Supported power control values are:`n`n- On`n- ForceOff`n-
GracefulRestart`n- GracefulShutdown"

return

```

```

}

if ($get_power_state_only -eq "n")
{
return
}

# POST command to set the new server power state. For POST commands, you must
create a hash table which must be converted to JSON format and compressed. This
behavior must be duplicated for all POST commands when creating the hash table.

$JsonBody = @{"ResetType" = $power_request_value
} | ConvertTo-Json -Compress

# Create the URI and POST command to change the server power state.
$u4 =
"https://$idrac_ip/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.Re
set"
$result1 = Invoke-WebRequest -Uri $u4 -Credential $credential -Method Post -Body
$JsonBody -ContentType 'application/json'

# Check if the POST command returned the correct status code
if ($result1.StatusCode -eq 204)
{
Write-Host
[String]::Format("- PASS, statuscode {0} returned, power control operation
success for: {1}", $result1.StatusCode, $power_request_value)
Start-Sleep 3
}
else
{
[String]::Format("- FAIL, statuscode {0} returned", $result1.StatusCode)
return
}

return
}

```

## Conclusion

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 Microsoft PowerShell, administrators can script access to the iDRAC REST API with Redfish, enabling the automation of server lifecycle functions including inventory, configuration, update, monitoring, and retirement, or repurposing. 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 business operations.

## References

- 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](http://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.0.1.pdf)
- The iDRAC with Lifecycle Controller on the Dell TechCenter provides access to product documents, technical white papers, and how-to videos: <http://en.community.dell.com/techcenter/systems-management/w/wiki/3204>
- Github repository for iDRAC REST API with Redfish support for Microsoft PowerShell and Python <https://github.com/dell/iDRAC-Redfish-Scripting>
- Open source version of Microsoft PowerShell <https://github.com/PowerShell/PowerShell>