# CONTENTS

1. Introduction ........................................................................................................... 10
2. References ................................................................................................................ 10
3. Overview .................................................................................................................... 12
   3.1 Format for WinRM CLI Examples in Document ..................................................... 12
   3.2 WS-Man Security & Time Parameters .................................................................. 13
      3.2.1 Encryption Certificate Security ................................................................... 13
      3.2.2 Handling invalid responses from WSMAN commands .................................. 13
      3.2.3 Improving WinRM Enumeration Performance ............................................. 14
      3.2.4 Specifying StartTime, Until Time, and TIME_NOW Parameters ................. 14
      3.2.5 Return Values ............................................................................................. 15
      3.2.6 Glossary ...................................................................................................... 15
4. Discovery .................................................................................................................... 16
   4.1 Discovering Web Service Capability .................................................................... 16
   4.2 Discovering what Profiles are Implemented ....................................................... 16
   4.3 Discovering Implementation Namespace ............................................................ 17
5. Managing iDRAC Local User Accounts ................................................................. 18
   5.1 Description of iDRAC Attributes vs Standard DMTF Model .............................. 18
   5.2 Account Inventory (using iDRAC Attributes) ....................................................... 18
      5.2.1 Account and Capabilities (using iDRAC Attributes) .................................... 18
      5.2.2 Privilege and Capabilities (using iDRAC Attributes) ................................... 19
   5.3 Manage Account Settings (using iDRAC Attributes) .......................................... 20
      5.3.1 Modify User Name (using iDRAC Attributes) .............................................. 20
      5.3.2 Modify Password (using iDRAC Attributes) .................................................. 21
      5.3.3 Modify Account State (using iDRAC Attributes) ......................................... 22
      5.3.4 Modify User Privilege (using iDRAC Attributes) ......................................... 23
   5.4 Account Inventory (using DMTF Model) .............................................................. 23
      5.4.1 Account and Capabilities (using DMTF Model) ........................................... 23
      5.4.2 Privilege and Capabilities (using DMTF Model) ............................................ 26
   5.5 Manage Account Settings (using DMTF Model) .................................................. 28
      5.5.1 Modify User Name (using DMTF Model) ....................................................... 28
      5.5.2 Modify Password (using DMTF Model) ......................................................... 31
      5.5.3 Modify Account State (using DMTF Model) .................................................. 31
      5.5.4 Modify User Privilege (using DMTF Model) .................................................. 32
6. Firmware Inventory .................................................................................................. 33
   6.1 Software Inventory Profile Specification ............................................................ 33
   6.2 Remote Inventory Method Invocation – Get Software Inventory ......................... 33
7. Firmware Update ...................................................................................................... 35
   7.1 Software Update Profile Specification ............................................................... 35
   7.2 “Rollback” Firmware .......................................................................................... 35
      7.2.1 Request “Rollback” Image ........................................................................... 35
7.2.2 Create Reboot Job ...........................................................................................................35
7.2.3 Schedule Update Jobs ....................................................................................................35
7.2.4 Monitor Update Jobs ......................................................................................................35
7.3 BIOS Firmware “Rollback” ...............................................................................................36
7.4 NIC Firmware “Rollback” ..................................................................................................37
7.5 Update from Network Source ...........................................................................................39
  7.5.1 Request Update Download ..........................................................................................39
  7.5.2 Monitor Download Status ..........................................................................................39
  7.5.3 Reboot to Perform Update .........................................................................................39
  7.5.4 Wait for Job Completion .............................................................................................39
  7.5.5 Delete Job ..................................................................................................................40
7.6 Update NICs from HTTP, CIFS Share, NFS share, TFTP, or FTP ...................................40
7.7 Update BIOS from HTTP, CIFS Share, NFS share, TFTP, or FTP .................................42
7.8 CreateRebootJob() ...........................................................................................................44
8 Power State Management ....................................................................................................45
  8.1 Description of Base Server vs Power State Management Methods ...............................45
  8.2 Get Power State ..............................................................................................................45
    8.2.1 Base Server Method .................................................................................................45
    8.2.2 Power State Management Method ..........................................................................46
  8.3 Get Power Control Capabilities .....................................................................................47
    8.3.1 Base Server Method .................................................................................................47
    8.3.2 Power State Management Method ..........................................................................48
  8.4 Power Control ................................................................................................................49
    8.4.1 Base Server Method .................................................................................................49
    8.4.2 Power State Management Method ..........................................................................50
9 Hardware Inventory ............................................................................................................51
  9.1 Power Supply Inventory ..................................................................................................51
  9.2 Fan Inventory ..................................................................................................................52
  9.3 Memory Inventory ..........................................................................................................53
  9.4 CPU Inventory ................................................................................................................54
  9.5 iDRAC Card Inventory ...................................................................................................55
  9.6 PCI Device Inventory ......................................................................................................56
  9.7 Video Inventory ...............................................................................................................57
  9.8 VFlash SD Card Inventory ...............................................................................................58
  9.9 NIC Inventory & Configuration .....................................................................................58
  9.10 RAID Inventory & Configuration ..................................................................................60
  9.11 BIOS Inventory & Configuration ..................................................................................62
  9.12 System Inventory (including CSI0R attribute) ...............................................................63
10 Job Control Management ..................................................................................................65
  10.1 Description of Job Management ...................................................................................65
  10.2 Remote Job Control Examples .....................................................................................65
    10.2.1 Setup Job Queue .....................................................................................................65
10.2.2 Delete Job Queue ................................................................. .66
10.2.3 List Jobs in Job Store ......................................................... .67
11 Operating System Deployment .................................................... .69
  11.1 OS Deployment Profile Implementation Conformance ..................... .69
  11.2 Checking OS Deployment Service Availability ............................. .69
  11.3 OS Deployment Method Invocation Examples ............................... .69
      11.3.1 Get Driver Pack Information ......................................... .70
      11.3.2 Unpack Selected Drivers and Attach to Host OS as USB Device ... .71
      11.3.3 Detach Emulated USB Device Containing Drivers .................. .72
      11.3.4 Unpack Selected Drivers and Copy to Network Share ............... .72
      11.3.5 Check Job Status ....................................................... .73
      11.3.6 Boot to Network ISO .................................................. .74
      11.3.7 Detach Network ISO USB Device ...................................... .75
      11.3.8 Boot To PXE ................................................................ .76
      11.3.9 Get Host MAC Address Information ................................... .77
      11.3.10 Download ISO to VFlash .............................................. .77
      11.3.11 Boot to ISO from VFlash .............................................. .78
      11.3.12 Delete ISO from VFlash ............................................... .79
      11.3.13 Detach ISO from VFlash ............................................... .80
      11.3.14 Connect Network ISO Image .......................................... .80
      11.3.15 Disconnect Network ISO Image ....................................... .81
      11.3.16 Skip ISO Image Boot ................................................... .82
      11.3.17 Get Network ISO Image Connection Information ................... .83
      11.3.18 Connect RFS ISO Image ............................................... .83
      11.3.19 Disconnect RFS ISO Image ............................................ .84
      11.3.20 Get RFS ISO Image Connection Information ......................... .85
      11.3.21 Boot To Hard Drive (HD) .............................................. .85
      11.3.22 Configurable Boot to Network ISO .................................... .86
12 Lifecycle Controller Management Profile ......................................... .87
  12.1 Collect System Inventory on Restart (CSIOR) ............................... .87
  12.2 Part Replacement Configuration and Management ......................... .89
      12.2.1 Create Config Job ...................................................... .89
      12.2.2 Get LC Config Job Status ............................................. .90
      12.2.3 List All LC Jobs ........................................................ .90
      12.2.4 Get CSIOR Component Configuration Recovery (CCR) Attribute ... .91
      12.2.5 Get Part Firmware Update Attribute ................................. .92
  12.3 Re-Initiate Auto-Discovery Client ............................................ .92
  12.4 Clear or Set Provisioning Server ............................................ .93
  12.5 Check VFlash License Enablement ........................................... .95
  12.6 Download Server Public Key ................................................ .95
  12.7 Download Client Certificates ................................................ .96
  12.8 Delete Auto-Discovery Client Certificates ................................... .97
12.9 Set Public Certificates ........................................................................................................... 98
12.10 Set iDRAC Certificate and Private Key ................................................................................... 99
12.11 Delete Auto-Discovery Server Public Key ............................................................................. 100
12.12 Insert Comment in Lifecycle Controller Log ......................................................................... 100
12.13 Export Lifecycle Controller Log ............................................................................................ 101
12.14 Export Hardware Inventory from Lifecycle Controller ......................................................... 102
12.15 Export Factory Configuration .................................................................................................. 103
12.16 System Decommission ........................................................................................................... 104
12.17 Get Remote Services API Status ............................................................................................ 105
12.18 Export System Configuration .................................................................................................. 105
12.19 Import System Configuration ................................................................................................. 106
13 VFlash SD Card Management ...................................................................................................... 107
   13.1 Listing the SD Card Partitions ............................................................................................... 108
   13.2 Initialize the Virtual Flash Media .......................................................................................... 108
      13.2.1 Get VFlash SD Card Inventory ....................................................................................... 109
      13.2.2 Initialize / Format Media ................................................................................................ 109
      13.2.3 Verify Initialization / Formatting ....................................................................................... 110
   13.3 Enable/Disable VFlash using VFlash State Change ............................................................... 111
   13.4 Create Partition ..................................................................................................................... 111
   13.5 Create Partition using Image .................................................................................................. 113
   13.6 Delete Partition ...................................................................................................................... 115
   13.7 Format Partition .................................................................................................................... 115
   13.8 Modify Partition ..................................................................................................................... 117
   13.9 Attach Partition ..................................................................................................................... 117
   13.10 Detach Partition ................................................................................................................... 118
   13.11 Export Data from Partition .................................................................................................. 119
14 Boot Control Configuration Management .................................................................................... 121
   14.1 Listing the Boot Inventory-ConfigSetting Class ..................................................................... 121
   14.2 Getting a Boot ConfigSetting Instance .................................................................................... 122
   14.3 Listing the Boot Inventory-SourceSetting Class .................................................................... 123
   14.4 Changing the Boot Order by InstanceID-ChangeBootOrderByInstanceID() ......................... 123
   14.5 Enable or Disable the Boot Source-ChangeBootSourceState() ............................................ 124
15 NIC/CNA Card Management ......................................................................................................... 125
   15.1 Listing the NIC/CNA Inventory-Enumeration Class ............................................................... 125
   15.2 Listing the NIC/CNA Inventory-String Class .......................................................................... 126
   15.3 Listing the CNA Inventory-Integer Class ................................................................................. 129
   15.4 Listing the CNA Inventory-NICView Class ............................................................................. 130
   15.5 Listing the CNA Inventory-NICCapabilities Class ................................................................. 132
   15.6 Listing the CNA Inventory- NICStatistics Class ..................................................................... 133
   15.7 Applying the Pending Values for CNA-CreateTargetedConfigJob() ..................................... 134
   15.8Deleting the Pending Values for CNA-DeletePendingConfiguration() .................................... 135
   15.9 Getting the CNA Enumeration Instance . ................................................................................. 136
15.10 Setting the IscsiOffloadMode Attribute ......................................................... 136
15.11 Setting the MaxBandwidth Attribute .............................................................. 138
15.12 Setting the VirtMacAddr Attribute ..................................................................... 139
15.13 Setting the LegacyBootProto Attribute ............................................................ 140
15.14 Setting CNA LAN Modes .................................................................................. 141
15.15 Setting the iSCSI Boot Target ........................................................................... 142
15.16 Setting the FCoE Boot Target ........................................................................... 143

16 RAID Storage Management .................................................................................. 144
16.1 Listing the RAID Inventory-Enumeration Class .................................................. 145
16.2 Getting a RAID Enumeration Instance .............................................................. 146
16.3 Listing the RAID Inventory-Integer Class ......................................................... 146
16.4 Getting a RAID Integer Instance ...................................................................... 148
16.5 Listing the RAID Inventory-String Class ............................................................ 148
16.6 Getting a RAID String Instance ....................................................................... 149
16.7 Listing the RAID Inventory-ControllerView Class ............................................. 150
16.8 Getting a RAID ControllerView Instance ......................................................... 151
16.9 Listing the RAID Inventory-PhysicalDiskView Class ......................................... 152
16.10 Listing the RAID VirtualDiskView Inventory .................................................. 153
16.11 Listing the RAID EnclosureView Inventory ...................................................... 155
16.12 Reset Configuration-ResetConfig() .................................................................. 155
16.13 Clearing the Foreign Configuration-ClearForeignConfig() .............................. 156
16.14 Applying the Pending Values for RAID-CreateTargetedConfigJob() .............. 157
16.15 Deleting the Pending Values for RAID-DeletePendingConfiguration() ............. 158

16.16 Managing Hot Spare ...................................................................................... 159
16.16.1 Determining Potential Disks-GetDHSDisks() ................................................ 159
16.16.2 Assigning the Hot Spare-AssignSpare() ....................................................... 159
16.16.3 Unassigning the Hot Spare-UnassignSpare() .............................................. 161

16.17 Managing Keys for Self Encrypting Drives ...................................................... 161
16.17.1 Setting the Key-SetControllerKey() .............................................................. 161
16.17.2 Locking the Virtual Disk-LockVirtualDisk() ............................................... 162
16.17.3 Locking the Controller with a Key-EnableControllerEncryption() .............. 163
16.17.4 Rekeying the Controller-ReKey() ................................................................. 164
16.17.5 Removing the Key-RemoveControllerKey() ............................................... 166

16.18 Managing Virtual Disk ................................................................................... 166
16.18.1 Getting the Available RAID levels-GetRAIDLevels() .................................... 166
16.18.2 Getting the Available Disks-GetAvailableDisks() ....................................... 168
16.18.3 Checking the Create VD Parameters Validity-CheckVDValues() .................. 169
16.18.4 Creating a Single Virtual Disk-CreateVirtualDisk() ....................................... 170
16.18.5 Creating a Sliced Virtual Disk-CreateVirtualDisk() ..................................... 173
16.18.6 Creating a Cachecade Virtual Disk-CreateVirtualDisk() ............................ 176
16.18.7 Deleting a Virtual Disk-DeleteVirtualDisk() ................................................ 178

16.19 Setting Controller Attributes ..................................................................... 179
16.19.1 Changing the Value of a RAID Controller Enumeration Attribute ..........179
16.19.2 Changing Multiple Values of RAID Controller Enumeration Attributes.....179
16.19.3 Changing the Value of a RAID Controller Integer Attribute ................180
16.19.4 Changing Multiple Values of RAID Controller Integer Attributes.........181
16.20 Convert Physical Disks to RAID-ConvertToRAID() ................................182
16.21 Convert Physical Disks to Non RAID-ConvertToNonRAID()......................183
17 Managing BIOS Configuration..................................................................183
  17.1 Listing the BIOS Inventory-Enumeration Class ......................................183
  17.2 Getting a BIOS Enumeration Instance ..................................................185
  17.3 Changing the BIOS BootMode-SetAttribute() .......................................185
  17.4 Setting Multiple BIOS BootMode Parameters .......................................186
  17.5 Listing the BIOS Inventory-Integer Class .............................................187
  17.6 Listing the BIOS Inventory-String Class ..............................................187
  17.7 Applying the Pending Values for BIOS & Boot-CreateTargetedConfigJob() ....188
  17.8 Deleting the Pending Values for BIOS & Boot-DeletePendingConfiguration() ....189
  17.9 Managing BIOS Passwords ..................................................................190
    17.9.1 Setting the BIOS Password ............................................................190
    17.9.2 Create Target Configuration Job ....................................................192
    17.9.3 Monitor Set BIOS Password Status ..............................................192
  17.10 Listing the BIOS Inventory-Password Class .........................................192
18 Exporting and Importing Server Profile......................................................193
  18.1 Exporting Server Profile ......................................................................194
    18.1.1 Exporting Server Profile to iDRAC vFlash Card-BackupImage() ............194
    18.1.2 Exporting Server Profile to NFS Share-BackupImage() .......................194
    18.1.3 Exporting Server Profile to CIFS Share-BackupImage() .....................195
    18.1.4 Monitoring Export status ..............................................................196
  18.2 Importing Server Profile ......................................................................197
    18.2.1 Importing Server Profile from iDRAC vFlash Card-RestoreImage() ........197
    18.2.2 Importing Server Profile from NFS share-RestoreImage() ....................197
    18.2.3 Importing Server Profile from CIFS share-RestoreImage() ...................198
    18.2.4 Monitoring Import Status .............................................................199
19 iDRAC Configuration..............................................................................200
  19.1 Listing the iDRAC Card Inventory-Enumeration Class .............................200
  19.2 Getting an iDRAC Card Enumeration Instance .......................................201
  19.3 Listing the iDRAC Card Inventory-Enumeration Class using groupId ...........202
  19.4 Applying the Attributes and Polling Job Completion ............................204
    19.4.1 Changing iDRAC Values-ApplyAttributes() (Immediate) .....................204
    19.4.2 Polling Job Completion ..................................................................205
    19.4.3 Set Attribute Verification ................................................................206
  19.5 Listing the iDRAC Card Inventory-Integer Class ....................................207
  19.6 Listing the iDRAC Card Inventory-Integer Class using groupId ...............208
  19.7 Listing the iDRAC Card Inventory-String Class .....................................209
19.8 Listing the iDRAC Card Inventory-String Class using *groupId* ........................................ 210
19.9 Changing the iDRAC IPChange Notification ................................................................. 212
    19.9.1 Getting the Current iDRAC IPChange State ..................................................... 212
    19.9.2 Setting the iDRAC IPChange Notification-SetAttribute() .................................. 212
20 Remote Service Status ........................................................................................................ 213
20.1 Getting Remote Service Status ....................................................................................... 213
20.2 Restarting Remote Service Status ................................................................................... 215
21 System Information ............................................................................................................ 216
21.1 Listing the System Inventory-SystemView Class ....................................................... 216
22 Sensor Information ............................................................................................................ 217
22.1 Listing the Sensors Inventory-PSNumericSensor Class ................................................. 217
23 Managing Fiber Channel (FC) Configuration ................................................................... 219
23.1 Listing the FC Inventory-Attribute Class ..................................................................... 219
23.2 Listing the FC Inventory-Statistics Class ..................................................................... 220
23.3 Listing the FC Inventory-String Class .......................................................................... 221
23.4 Listing the FC Inventory-Integer Class ......................................................................... 222
23.5 Listing the FC Inventory-Enumeration Class ............................................................... 222
23.6 Changing the FC Attributes-SetAttribute() .............................................................. 223
23.7 Applying the Pending Values for FC-CreateTargetedConfigJob() .............................. 223
23.8 Deleting the Pending Values for FC-DeletePendingConfiguration() .............................. 225
23.9 Listing the FC Views ..................................................................................................... 225
1 Introduction

This document serves as a guideline for utilizing the functionality available from embedded Lifecycle Controller Remote Enablement Web Services interfaces. The purpose of this document is to provide information and examples for utilizing the Web services for Management (WS-Man) management protocol using Windows WinRM and open source WSMANCLI command line utilities. Examples and invocation information is provided for the following functionality.

- Inventory for BIOS, component firmware and embedded software
- Update of BIOS, component firmware and embedded software
- Job Control of update tasks
- Enhancement of Operating System Deployment using VFlash SD Card
- Enhancement of Discovery and Handshake from LifeCycle Controller 1.x
- Raid configuration management
- iDRAC Inventory and configuration features
- NIC configuration management
- Boot configuration management
- BIOS configuration management

The target audience for this document is application and script writers that want to utilize the remote management capabilities using WS-Man protocol available from Dell Lifecycle Controller.

2 References

1 Dell 12th Generation PowerEdge Server Resources: http://www.delltechcenter.com/12thGen

2 Dell CIM Profiles: http://www.delltechcenter.com/page/DCIM.Library.Profile

3 Managed Object Format (MOF) files http://www.delltechcenter.com/page/DCIM.Library.MOF


5 Openwsman CLI: http://www.openwsman.org/project/wsmancli

6 DMTF Common Information Model (CIM) Infrastructure Specification (DSP0004): http://www.dmtf.org/standards/published_documents/DSP0004_2.5.0.pdf
List of PCI IDs:

http://pciids.sourceforge.net/pci.ids
3 Overview

The remote interface guidelines provided in this document are illustrated by command line examples of the WS-MAN protocol Web services APIs that expose the remote management capabilities of the Dell Lifecycle Controller. The command line examples are from the Microsoft® Windows® and Linux environments using WinRM and WSMANCLI respectively. The Lifecycle Controller remote management capabilities are organized by management domain and documented in Dell CIM Profile specifications. The remote enablement feature for Lifecycle Controller 2.0 provides the following capabilities:

- Remotely get inventory of the BIOS, component firmware, and embedded software including version information of both the installed as well as available cached versions
- Remote update of BIOS, component firmware, Diagnostic content, DRAC content, driver pack, power supplies from remotely located Dell Update Packages or cached images located in the Lifecycle Controller
- Remotely schedule and track the status of update tasks (jobs)
- Remotely manage the Part Replacement feature by allowing retrieving and setting auto update and auto system inventory sync
- Enable re-initiation of Lifecycle Controller Auto-Discovery feature
- Enhancement of Operation System Deployment capabilities by supporting the downloading of an ISO image to a Dell VFlash SD Card and booting to the ISO image on the VFlash SD Card
- NIC configuration enables the ability to get and set NIC attributes that are configurable using NIC Option ROM or NIC UEFI HII.
- Remote RAID configuration allows users to remotely query and configure the Hardware Raid of the system
- Multiple HW Inventory views allows users to remote query the inventory of Hardware

3.1 Format for WinRM CLI Examples in Document

The examples of WinRM and WSMANCLI command line invocations in this document are formatted for readability and often span multiple lines in the document. In actual use, scripted or hand-typed invocations are contained on one line. The examples also use substitute values for the target iDRAC IP address, username (with ExecuteServerCommand privilege), password and other site specific information. Actual use of these examples would require using values for IP Address, username and password, etc. that are valid. These values are represented in the examples as follows:

Target iDRAC IP address = [IPADDRESS]

iDRAC Username = [USER]

iDRAC Password = [PASSWORD]

Additional substitute values are used in some of the examples and are described in the specific example.
The following example is typical of the formatting used in this document:

**EXAMPLE:**

```plaintext
winrm e cimv2/root/dcim/DCIM_OSDeploymentService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-encoding:utf-8 -a:basic
```

### 3.2 WS-Man Security & Time Parameters

#### 3.2.1 Encryption Certificate Security

For the WinRM examples provided in this document, the strict checks of certificates such as matching of CNs (Common Names) and verification with the actual CA (Certificate Authority) of the certificate of the WS-Management protocol HTTPS encryption certificate is assumed to be already configured and enabled. To disable the strict certificate checking, add the following command line options to all WinRM examples: `–skipCACheck` and `–skipCNCheck`.

Additionally, the following error may result if the end point does not support this feature. Use the switch `–skiprevocationcheck` to bypass this error.

```plaintext
WSManFault
Message = The server certificate on the destination computer (10.35.0.232:443) has the following errors:

The SSL certificate could not be checked for revocation. The server used to check for revocation might be unreachable.

Refer to the WinRM documentation and related documentation for directions on setting up encryption certificates for WinRM and executing WinRM invocations using full security capabilities. Refer to the Lifecycle Controller User Guide for directions on configuring different encryption certificates for the iDRAC Web server. Dell recommends that the full security and encryption capabilities of the WS-Management protocol is used for production level utilization of the Lifecycle Controller Web services interfaces.
```

#### 3.2.2 Handling invalid responses from WSMAN commands

- Check the network connection to make sure that the system is connected
- Check the WSMAN syntax to ensure there are no typos in the command line
- Check if there are other WSMAN commands sending from other systems
- Wait for a few seconds and re-try the WSMAN command
3.2.3 Improving WinRM Enumeration Performance

When an enumeration command is executed, the default WinRM configuration gets only 20 instances at a time and therefore slows down the system drastically. Changing the WinRM configuration to allow a greater number, such as 50, will reduce the time taken by the enumeration operations.

Execute the following command to get instances in groups of up to 50.

```bash
winrm set winrm/config @{MaxBatchItems="50"}
```

Additionally, increasing the allotted maximum envelope size and timeout can also increase performance.

```bash
winrm set winrm/config @{MaxEnvelopeSizekb="150"}
winrm set winrm/config @{MaxTimeoutms ="60000"}
```

Other optional WinRM configuration commands are listed below for convenience. To get the current WinRM configuration settings, execute the following command.

```bash
winrm g winrm/config
```

By default, the client computer requires encrypted network traffic. To allow the client computer to request unencrypted traffic, execute the following command:

```bash
winrm s winrm/config/Client @{AllowUnencrypted="true"}
```

`TrustedHosts` is an array that specifies the list of remote computers that are trusted. Other computers in a workgroup or computers in a different domain should be added to this list.

Note: The computers in the `TrustedHosts` list are not authenticated.

Execute the following command to allow all computers to be included in `TrustedHosts`.

```bash
winrm s winrm/config/Client @{TrustedHosts="*"}
```

Basic authentication is a scheme in which the user name and password are sent in clear text to the server or proxy. This method is the least secure method of authentication. The default is True.

Execute the following command to set client computer to use Basic authentication.

```bash
winrm s winrm/config/Client/Auth @{Basic="true"}
```

3.2.4 Specifying `StartTime`, `Until Time`, and `TIME_NOW` Parameters

The several methods that attach a virtual USB device to the target system accept a `StartTime` and `Until` parameter. The parameter data type is CIM date-time. If the `StartTime` parameter is null the action will not be started. If the `Until` parameter is null, the default value will be 17 hours. The date-time data type is defined in the CIM Infrastructure Specification\(^4\) as:

```
dddddddddddhhmmss.mmmmmm
```

Where:
• dddddddd is the number of days
• hh is the remaining number of hours
• mm is the remaining number of minutes
• ss is the remaining number of seconds
• mmmm is the remaining number of microseconds

The Lifecycle controller firmware update, and set attribute related methods that require a date time parameter, use the form YYYYMMDDhhmmss (Eg. 20090930112030). The user is expected to enter the date and time in this format for all Lifecycle Controller updates and set attribute tasks. TIME_NOW is a special value that represents “running the tasks immediately”.

3.2.5 Return Values
Many of the methods in this document have the following possible return values. They are summarized here for convenience.

0 = Success
1 = Not Supported
2 = Failed
4096 = Job Created

3.2.6 Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIOS</td>
<td>Basic Input / Output System</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
<tr>
<td>iDRAC</td>
<td>Integrated DELL Remote Access Controller</td>
</tr>
<tr>
<td>IPL</td>
<td>Initial Program Load</td>
</tr>
<tr>
<td>DUP</td>
<td>Dell Update Package</td>
</tr>
<tr>
<td>MOF</td>
<td>Managed Object File</td>
</tr>
<tr>
<td>CIM</td>
<td>Common Information Model</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Controller</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Array of Independent Disks</td>
</tr>
<tr>
<td>FQDD</td>
<td>Fully Qualified Device Description</td>
</tr>
<tr>
<td>UEFI</td>
<td>Unified Extensible Firmware Interface</td>
</tr>
<tr>
<td>AMEA</td>
<td>Advanced Management Enablement Adapter</td>
</tr>
<tr>
<td>HII</td>
<td>Human Interface Infrastructure</td>
</tr>
<tr>
<td>WSMAN</td>
<td>WS-Management is a specification of a SOAP-based protocol for the</td>
</tr>
<tr>
<td></td>
<td>management of servers, devices, applications and more</td>
</tr>
</tbody>
</table>
4  Discovery

4.1  Discovering Web Service Capability
Determine if the target system supports the WinRM interface using the ‘identify’ command.

Profiles:  
http://www.dmtf.org/sites/default/files/standards/documents/DSP0217_2.0.0.pdf

EXAMPLE:

winrm identify
-u:[USER]  -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman  -SkipCNcheck -SkipCAcheck
-encoding:utf-8  -a:basic

OUTPUT:

IdentifyResponse
  ProductVendor = Openwsman Project
  ProductVersion = 2.2.4

4.2  Discovering what Profiles are Implemented
Implemented profiles are advertised using the class CIM_RegisteredProfile. Enumerate this class in the “root/interop” CIM namespace.

Profiles:  
http://www.dmtf.org/sites/default/files/standards/documents/DSP1033_1.0.0.pdf

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_RegisteredProfile?__cimnamespace=root/interop
-u:[USER]  -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman  -SkipCNcheck -SkipCAcheck
-encoding:utf-8  -a:basic

OUTPUT:

DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify, Interop Namespace
  AdvertiseTypes = 1, 1
  InstanceID = DCIM:Memory:1.0.0
  OtherRegisteredOrganization = DCIM
  RegisteredName = Memory
  RegisteredOrganization = 1
  RegisteredVersion = 1.0.0
  ...
  DCIM_RegisteredProfile
AdvertiseTypeDescriptions = WS-Identify
AdvertiseTypes = 1
Caption = null
Description = null
ElementName = null
InstanceID = DCIM:CSRegisteredProfile:1
OtherRegisteredOrganization = null
RegisteredName = Base Server
RegisteredOrganization = 2
RegisteredVersion = 1.0.0

The above example shows that the DMTF Base Server profile version 1.0.0 is implemented.

4.3 Discovering Implementation Namespace
The implementation CIM namespace may be discovered from the interop (root/interop) CIM namespace using the class CIM_ElementConformsToProfile that associates an instance of CIMRegisteredProfile class with an instance of CIM_ComputerSystem class.

Profiles: n/a

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/* -dialect:association -associations -filter:
{object=DCIM_ComputerSystem?CreationClassName=DCIM_ComputerSystem+Name=srv:system+__cimnamespace=root/dcim}
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -encoding:utf-8 -a:basic
-SkipCNcheck -SkipCAcheck

OUTPUT:

DCIM_CSRoleLimitedToTarget
  DefiningRole
    Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    ReferenceParameters
      ResourceURI = http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_Role
      SelectorSet
        Selector: CreationClassName = DCIM_Role, Name = DCIM:Role:9, __cimnamespace = root/dcim
  TargetElement
    Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    ReferenceParameters
      ResourceURI = anonymous
      SelectorSet
        Selector: CreationClassName = DCIM_ComputerSystem, Name = srv:system, __cimnamespace = root/dcim

17
Managing iDRAC Local User Accounts

5.1 Description of iDRAC Attributes vs Standard DMTF Model
The iDRAC user account management data model is represented by both DMTF and Dell Profiles. Both models are currently offered. The DMTF Profiles for Simple Identity Management and Role Based Authorization represent iDRAC user accounts and privileges. The DMTF data model is complex and typically requires multiple transactions to accomplish simple operations such as specifying a username and password or giving a user account admin privileges. For this reason, LC also offers a Dell data model for managing iDRAC user accounts that is based on an attribute model. The DCIM iDRAC Card Profile specifies the attributes for each user account name, password, and privilege. The iDRAC has 15 local user account that can be managed.

5.2 Account Inventory (using iDRAC Attributes)
The list of user accounts may be retrieved by enumerating the DCIM_iDRACCard classes. The class provides the user account name and enabled state properties.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

5.2.1 Account and Capabilities (using iDRAC Attributes)
Enumerating the DCIM_iDRACCardEnumeration class, Section 19.1, and parsing the output for the attribute AttributeDisplayName = User Admin Enable, will display all of the 16 possible user accounts and their respective status.

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardEnumeration
-u:[USER] -p:[PASSWORD]
Enumerating the `DCIM_iDRACCardEnumeration` class, Section 19.1, and parsing the output for the attribute `AttributeDisplayName = User Admin IPMI LAN(or Serial) Privilege`, will display all of the 16 possible user accounts and their respective status.

**EXAMPLE:**

```
DCIM_iDRACCardEnumeration
  AttributeDisplayName = User Admin IPMI LAN Privilege
  AttributeName = IpmiLanPrivilege
  CurrentValue = NoAccess
  DefaultValue = NoAccess
  Dependency = null
  DisplayOrder = 0
  FQDD = iDRAC.Embedded.1
  GroupDisplayName = Users
  GroupID = Users.1
  InstanceID = iDRAC.Embedded.1#Users.1#IpmiLanPrivilege
  IsReadOnly = true
  PossibleValues = Disabled, Enabled
```

Account **Disabled** as displayed in `CurrentValue` attribute for **Users.1**

```
DCIM_iDRACCardEnumeration
  AttributeDisplayName = User Admin IPMI LAN Privilege
  AttributeName = IpmiLanPrivilege
  CurrentValue = Enabled
  DefaultValue = Enabled
  Dependency = null
  DisplayOrder = 0
  FQDD = iDRAC.Embedded.1
  GroupDisplayName = Users
  GroupID = Users.2
  InstanceID = iDRAC.Embedded.1#Users.2#IpmiLanPrivilege
  IsReadOnly = false
  PossibleValues = Disabled, Enabled
```

Account **Enabled** as displayed in `CurrentValue` attribute for **Users.2**
InstanceID = iDRAC.Embedded.1#Users.1#IpmiLanPrivilege
IsReadOnly = true
PossibleValues = User, Operator, Administrator, NoAccess

DCIM_iDRACCardEnumeration
    AttributeDisplayName = User Admin IPMI Serial Privilege
    AttributeName = IpmiSerialPrivilege
    CurrentValue = NoAccess
    DefaultValue = NoAccess
    Dependency = null
    DisplayOrder = 0
    FQDD = iDRAC.Embedded.1
    GroupDisplayName = Users
    GroupID = Users.1
    InstanceID = iDRAC.Embedded.1#Users.1#IpmiSerialPrivilege
    IsReadOnly = true
    PossibleValues = User, Operator, Administrator, NoAccess.

5.3 Manage Account Settings (using iDRAC Attributes)
When the account setting capability allows, the user name of an account may be modified by invoking
the ApplyAttributes() method on the UserName property. Confirmation of successful user name or
password verification can be obtained by enumerating the DCIM_iDRACCardString class (Section 19.6).

5.3.1 Modify User Name (using iDRAC Attributes)

EXAMPLE:

```
winrm i ApplyAttributes http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_iDRACCardService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:iDRACCardService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file: DracCard_UserName.xml
```

The input file, DracCard_UserName.xml, is shown below:

```
  <p:Target>iDRAC.Embedded.1</p:Target>
  <p:AttributeName>Users.4#UserName</p:AttributeName>
  <p:AttributeValue>HELLO</p:AttributeValue>
</p:ApplyAttributes_INPUT>
```

OUTPUT:
When this method is executed, a `jobid` or an error message is returned.

```
ApplyAttributes_OUTPUT
   ReturnValue = 4096
   Job
      EndpointReference
         Address = https://127.0.0.1:443/wsman
      ReferenceParameters
      SelectorSet
         Selector: __cimnamespace = root/dcim,
         InstanceID = JID_001296571842
```

### 5.3.2 Modify Password (using iDRAC Attributes)

**EXAMPLE:**

```
winrm i ApplyAttributes http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardService
  ?SystemCreationClassName=DCIM_ComputerSystem
  +CreationClassName=DCIM_iDRACCardService
  +SystemName=DCIM:ComputerSystem
  +Name=DCIM:iDRACCardService
  -u:[USER] -p:[PASSWORD]
  -r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
  -encoding:utf-8 -a:basic -file:DracCard_Password.xml
```

The input file, `DracCard_Password.xml`, is shown below:

```
   <p:Target>iDRAC.Embedded.1</p:Target>
   <p:AttributeName>Users.4#Enable</p:AttributeName>
   <p:AttributeValue>Enabled</p:AttributeValue>
   <p:AttributeName>Users.4#Password</p:AttributeName>
   <p:AttributeValue>PWORDHERE</p:AttributeValue>
</p:ApplyAttributes_INPUT>
```

**OUTPUT:**

When this method is executed, a `jobid` or an error message is returned.

```
ApplyAttributes_OUTPUT
   ReturnValue = 4096
   Job
      EndpointReference
         Address = https://127.0.0.1:443/wsman
      ReferenceParameters
      SelectorSet
         Selector: __cimnamespace = root/dcim,
         InstanceID = JID_001296571842
```
5.3.3 Modify Account State (using iDRAC Attributes)

When the account setting capability allows, the user account may be enabled or disabled by invoking the method `ApplyAttributes()` method on the `Enable` property. Confirmation of the change can be obtained by enumerating the `DCIM_iDRACCardString` class (Section 19.6).

**EXAMPLE:**

```
winrm i ApplyAttributes http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_iDRACCardService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:iDRACCardService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -skipCNcheck -skipCAcheck
-encoding:utf8 -a:basic -file: DracCard_AccountChange.xml
```

The input file, `DracCard_AccountChange.xml`, is shown below:

```
  <p:Target>iDRAC.Embedded.1</p:Target>
  <p:AttributeName>Users.4#Enable</p:AttributeName>
  <p:AttributeValue>Enabled</p:AttributeValue>
  <p:AttributeName>Users.4#Password</p:AttributeName>
  <p:AttributeValue>PASSWORDHERE</p:AttributeValue>
</p:ApplyAttributes_INPUT>
```

**OUTPUT:**

When this method is executed, a `jobid` or an error message is returned.

```
ApplyAttributes_OUTPUT
  Job
    Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    ReferenceParameters
    SelectorSet
      Selector: InstanceID = JID_001296744532, __cimnamespace = root/dcim
      ReturnValue = 4096
```

The following error may result if the password has not initially been set to a value. The password may be set an initial value at the same time as the account is enabled by adding the `Users.4#Password` attribute name and corresponding attribute value, as shown above.

```
ApplyAttributes_OUTPUT
  Message = The User Password is not configured so cannot Enable the User or set values for IPMILan IPMISerial or User Admin Privilege
  MessageArguments = NULL
  MessageID = RAC023
  ReturnValue = 2
```
5.3.4 Modify User Privilege (using iDRAC Attributes)
When the account setting capability allows, the user privileges may be enabled or disabled by invoking the method ApplyAttributes() method on the Enable property. Confirmation of the change can be obtained by enumerating the DCIM_iDRACCardString class (Section 19.6).

EXAMPLE:

```
winrm i ApplyAttributes http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardService
/SystemCreationClassName=DCIM_ComputerSystem +CreationClassName=DCIM_iDRACCardService +SystemName=DCIM:ComputerSystem +Name=DCIM:iDRACCardService -u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -file: DracCard_PrivilegeChange.xml
```

The input file, DracCard_PrivilegeChange.xml, is shown below:

```xml
  <p:Target>iDRAC.Embedded.1</p:Target>
  <p:AttributeName>Users.4#IpmiLanPrivilege</p:AttributeName>
  <p:AttributeValue>Operator</p:AttributeValue>
</p:ApplyAttributes_INPUT>
```

OUTPUT:

When this method is executed, a jobid or an error message is returned.

```
ApplyAttributes_OUTPUT
  Job
    Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    ReferenceParameters
    SelectorSet
      Selector: InstanceID = JID_001296745342, __cimnamespace = root/dcim
    ReturnValue = 4096
```

5.4 Account Inventory (using DMTF Model)
The list of user accounts may be retrieved by enumerating the CIM_Account class. The class provides the user account name and EnabledState properties. The user account password is also included but it is a write-only property.

Profiles:

http://www.dmtf.org/sites/default/files/standards/documents/DSP1034_1.0.1.pdf
http://www.dmtf.org/sites/default/files/standards/documents/DSP1039_1.0.0.pdf

5.4.1 Account and Capabilities (using DMTF Model)
Example-A demonstrates standard output. Example-B demonstrates EPR mode output.
**EXAMPLE-A:**

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_Account
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT-A:**

```
DCIM_Account
  CreationClassName = DCIM_Account
  ElementName = DCIM Account
  EnabledDefault = 2
  EnabledState = 3
  Name = iDRAC.Embedded.1#Users.1
  OrganizationName = DCIM
  RequestedState = 0
  SystemCreationClassName = DCIM_SPComputerSystem
  SystemName = systemmc
  TransitioningToState = 12
  UserID = null
  UserPassword = null
```

```
DCIM_Account
  CreationClassName = DCIM_Account
  ElementName = DCIM Account
  EnabledDefault = 2
  EnabledState = 2
  Name = iDRAC.Embedded.1#Users.2
  OrganizationName = DCIM
  RequestedState = 0
  SystemCreationClassName = DCIM_SPComputerSystem
  SystemName = systemmc
  TransitioningToState = 12
  UserID = root
  UserPassword = null
```

**EXAMPLE-B:**

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_Account
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT-B:**

```
EndpointReference
  Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
  ReferenceParameters
    ResourceURI = http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_Account
    SelectorSet
```
Account setting capability is defined in the class `CIM_AccountManagementCapabilities` associated with the `CIM_Account` class instance. The ability to enable and disable an account is defined in the capability class `CIM_EnabledLogicalElementCapabilities` associated with the `CIM_Account` class.

To determine account setting capabilities:

1. Get the `CIM_Account` class instance of interest using EnumerateEPR mode.
2. Enumerate the associators of the `CIM_Account` instance and search for `CIM_AccountManagementService` class instance using EnumerateEPR mode.
3. Enumerate the associators of the `CIM_AccountManagementService` instance and search for `CIM_AccountManagementCapabilities` class instance.
4. One exception is account index 0. The first account is static and could not be set.

**OUTPUT-C:**

```
DCIM_LocalUserAccountManagementCapabilities
  ElementName = Local User Account Management Capabilities
  ElementNameEditSupported = false
  InstanceID = DCIM:LocalUserAccountManagementCapabilities:1
  MaxElementNameLen = 0
  OperationsSupported = 3
  SupportedAuthenticationMethod = 0, 1, 2

DCIM_IPMICLPAccountManagementCapabilities
  ElementName = IPMI/CLP Account Management Capabilities
  ElementNameEditSupported = false
  InstanceID = DCIM:IPMICLPAccountManagementCapabilities:1
  MaxElementNameLen = 0
  OperationsSupported = 3
```
To determine account state setting capabilities:

1. Get the CIM_Account class instance of interest using EnumerateEPR mode.
2. Enumerate the associators of the CIM_Account instance and search for CIM_EnabledLogicalElementCapabilities class instance.
3. The presence of “RequestedStatesSupported” determines which states could be set.
4. One exception is account index 0. The first account is static and could not be set.

OUTPUT-D:

DCIM_EnabledLogicalElementCapabilities
   ElementName = Account Capabilities
   ElementNameEditSupported = false
   InstanceID = DCIM_EnabledLogicalElementCapabilities:1
   MaxElementNameLen = 0
   RequestedStatesSupported = 2, 3

5.4.2 Privilege and Capabilities (using DMTF Model)

The account privilege assigned to a user is defined in the class CIM_Privilege associated with the CIM_Account class. The class contains a list of privileges granted to the user account.

Profiles:

http://www.dmtf.org/sites/default/files/standards/documents/DSP1034_1.0.1.pdf
http://www.dmtf.org/sites/default/files/standards/documents/DSP1039_1.0.0.pdf

To get the instance of CIM_Privilege for an account:

1. Get the CIM_Account class instance of interest using EnumerateEPR mode.
2. Enumerate the associators of the CIM_Account instance and search for CIM_Identity class instance using EnumerateEPR mode.
3. Enumerate the associators of the CIM_Identity instance and search for CIM_Role class instance using EnumerateEPR mode.
4. Enumerate the associators of the CIM_Role instance and search for CIM_Privilege class instance.

An alternative to the above method, you can retrieve the specific CIM_Privilege instance by enumerating the class directly with filter. This method is similar to the example used to retrieve CIM_Account.

EXAMPLE:
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/DCIM_LocalRolePrivilege
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman
-SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic

OUTPUT:

DCIM_LocalRolePrivilege
   Activities = null
   ActivityQualifiers = null
   ElementName = DCIM Local Privilege 1
   InstanceID = DCIM:Privilege:1
   PrivilegeGranted = true
   QualifierFormats = null
   RepresentsAuthorizationRights = false

DCIM_LocalRolePrivilege
   Activities = 7, 7, 7, 7, 7, 7, 7, 7, 7
   ActivityQualifiers = Login to DRAC, Configure DRAC, Configure Users, Clear Logs, Test Alerts, Execute Server Control Commands, Access Console Redirection, Access Virtual Media, Execute Diagnostic Commands
   ElementName = DCIM Local Privilege 2
   InstanceID = DCIM:Privilege:2
   PrivilegeGranted = true
   QualifierFormats = 9, 9, 9, 9, 9, 9, 9, 9, 9
   RepresentsAuthorizationRights = true

DCIM_LocalRolePrivilege
   Activities = null
   ActivityQualifiers = null

   ElementName = DCIM Local Privilege 3
   InstanceID = DCIM:Privilege:3
   PrivilegeGranted = true
   QualifierFormats = null
   RepresentsAuthorizationRights = false

Privilege setting capability is defined in the class CIM_RoleBasedManagementCapabilities associated with the CIM_Privilege class instance. This class contains the list of possible values used to assign privileges. Look for the property ActivityQualifiersSupported.

To determine privilege setting capabilities:

1. Acquire the class instance of CIM_Privilege of interest.
2. Enumerate the associators of the CIM_Privilege instance and search for CIM_RoleBasedAuthorizationService class instance using EnumerateEPR mode.
3. Enumerate the associators of the CIM_RoleBasedAuthorizationService instance and search for CIM_RoleBasedManagementCapabilities class instance using EnumerateEPR mode.

**OUTPUT:**

DCIM_LocalRoleBasedManagementCapabilities
  ActivitiesSupported = 7, 7, 7, 7, 7, 7, 7, 7, 7
  ActivityQualifiersSupported = Login to DRAC, Configure DRAC, Configure Users, Clear Logs, Execute Server Control Commands, Access Console Redirection, Access Virtual Media, Test Alerts, Execute Diagnostic Commands
  ElementName = Local Role Based Management Capabilities
  InstanceID = DCIM:LocalRoleBasedManagementCapabilities
  QualifierFormatsSupported = 9, 9, 9, 9, 9, 9, 9, 9, 9
  SharedPrivilegeSupported = false
  SupportedMethods = 8

DCIM_CLPRoleBasedManagementCapabilities
  ActivitiesSupported = null
  ActivityQualifiersSupported = null
  ElementName = CLP Role Based Management Capabilities
  InstanceID = DCIM:CLPRoleBasedManagementCapabilities
  QualifierFormatsSupported = null
  SharedPrivilegeSupported = false
  SupportedMethods = 6

DCIM_IPMIRoleBasedManagementCapabilities
  ActivitiesSupported = null
  ActivityQualifiersSupported = null
  ElementName = IPMI Role Based Management Capabilities
  InstanceID = DCIM:IPMIRoleBasedManagementCapabilities
  QualifierFormatsSupported = null
  SharedPrivilegeSupported = false
  SupportedMethods = 6

5.5 Manage Account Settings (using DMTF Model)

5.5.1 Modify User Name (using DMTF Model)
When the account setting capability allows, the user name of an account may be modified by issuing a set operation on the UserID property of the CIM_Account class instance. The set operation requires an instance reference. The instance reference may be retrieved by adding EnumerateEPR mode to enumerate or get of the class.

Profiles:

http://www.dmtf.org/sites/default/files/standards/documents/DSP1034_1.0.1.pdf
http://www.dmtf.org/sites/default/files/standards/documents/DSP1039_1.0.0.pdf
The steps below demonstrate how to set the user name and password for local accounts.

A) Enumerate CIM_Account with EPR to identify all possible instance information to be used in a subsequent put or set operations.

**EXAMPLE-A:**

```sh
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_Account
?__cimnamespace=root/dcim
-u:[USER] -p:[PASSWORD]
-r:http://[IPADDRESS]/wsman:443 -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -returntype:EPR
```

When this command is executed, a list of objects will be returned. Below is a snippet of the output.

**OUTPUT-A:**

```
EndpointReference
   Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
   ReferenceParameters
      ResourceURI = http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_Account
      SelectorSet
         Selector: __cimnamespace = root/dcim, Name = iDRAC.Embedded.1#Users.1,
         CreationClassName = DCIM_Account, SystemName = systemmc, SystemCreationClassName = DCIM_SPComputerSystem
```

```
EndpointReference
   Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
   ReferenceParameters
      ResourceURI = http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_Account
      SelectorSet
         Selector: __cimnamespace = root/dcim, Name = iDRAC.Embedded.1#Users.2,
         CreationClassName = DCIM_Account, SystemName = systemmc, SystemCreationClassName = DCIM_SPComputerSystem
```

B) Perform a ‘get’ on any instance from A) to ensure correctness of the URI.

**EXAMPLE-B:**

```sh
winrm g "http://schemas.dell.com/wbem/wscim/1/cim-schema/2/
DCIM_Account?__cimnamespace=root/dcim
+CreationClassName= DCIM_Account
+Name= iDRAC.Embedded.1#Users.16
+SystemCreationClassName=DCIM_SPComputerSystem
+SystemName=systemmc"
```
When this method is executed, the particular object will be returned. Below is the output.

**OUTPUT-B:**

```
DCIM_Account
  CreationClassName = DCIM_Account
  ElementName = DCIM Account
  EnabledDefault = 2
  EnabledState = 3
  Name = iDRAC.Embedded.1#Users.16
  OrganizationName = DCIM
  RequestedState = 0
  SystemCreationClassName = DCIM_SPComputerSystem
  SystemName = systemmc
  TransitioningToState = 12
  UserID = null
  UserPassword = null
```

C) If B) is successful, set the new values for the specified instance.

**EXAMPLE-C:**

```
winrm set "http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_Account?__cimnamespace=root/dcim" +CreationClassName= DCIM_Account +Name= iDRAC.Embedded.1#Users.16 +SystemCreationClassName=DCIM_SPComputerSystem +SystemName=systemmc -r:https://[IPADDRESS] -u:[USER] -p:[PASSWORD] -a:basic -encoding:utf-8 -SkipCACheck -SkipCNCheck
```

When this command is executed, the UserID will be displayed in the output. The UserPassword will be displayed as null when the account is disabled. After the account is enabled, it will be displayed as blank. The value of UserPassword will never be displayed.

**OUTPUT-C:**

```
DCIM_Account
  CreationClassName = DCIM_Account
  ElementName = DCIM Account
  EnabledDefault = 2
  EnabledState = 3
  Name = iDRAC.Embedded.1#Users.16
  OrganizationName = DCIM
```
RequestedState = 0
SystemCreationClassName = DCIM_SPComputerSystem
SystemName = systemmc
TransitioningToState = 12
UserID = testuser4
UserPassword = null
UserID = testuser4
UserPassword = testuser4

D) If the account specified is new or not yet enabled, it will not be accessible. Login as root in the UI and verify the user name is set correctly and enable it.

E) Logout of the UI. Logging in with new user name and password and be successful.

Possible responses:

1. A fault is returned which suggests a possible error in the request payload.
2. An empty response which suggests an error occurred while processing the request.
3. An instance of the class is returned where the property value is unchanged.
4. An instance of the class is returned where the property value is modified. The set is successful.
5. The property value may be blank as intended by the implementation for security. To determine success, try logging in with the new password. Ensure the account is enabled.

5.5.2 Modify Password (using DMTF Model)
When the account setting capability allows, the user password of an account may be modified by issuing a set operation on the UserPassword property of the CIM_Account class instance. The set operation requires an instance reference. The instance reference may be retrieved by adding EnumerateEPR mode to enumerate or get of the class.

NOTE: The profile defines this property as string array of type octet string. In this implementation, the password is a string of type clear text. The security concern is resolved by transmission of this information only through secure HTTPS communication.

Profiles:

http://www.dmtf.org/sites/default/files/standards/documents/DSP1034_1.0.1.pdf
http://www.dmtf.org/sites/default/files/standards/documents/DSP1039_1.0.0.pdf

See Section 5.5.1 for an implementation example.

5.5.3 Modify Account State (using DMTF Model)
When the account setting capability allows, the user account may be enabled or disabled by invoking the RequestStateChange() method of the CIM_Account class instance. The invoke operation requires
an instance reference. The instance reference may be retrieved by adding EnumerateEPR mode to enumerate or get of the class.

Profiles:

http://www.dmtf.org/sites/default/files/standards/documents/DSP1034_1.0.1.pdf
http://www.dmtf.org/sites/default/files/standards/documents/DSP1039_1.0.0.pdf

Replace “DCIM User 16” with the applicable user name and “2” with the desired request state.

Invoke RequestStateChange() with the following parameters and syntax:

EXAMPLE:

winrm invoke RequestStateChange "http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/DCIM_Account
?__cimnamespace=root/dcim
+CreationClassName=DCIM_Account
+Name= iDRAC.Embedded.1#Users.16
+SystemCreationClassName=DCIM_SPComputerSystem
+SystemName=systemmc"
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic @{RequestedState="2"} -skiprevocationcheck

OUTPUT:

RequestStateChange_OUTPUT
  ReturnValue = 0

Response status other than zero indicates failure and error message information may be provided.

5.5.4 Modify User Privilege (using DMTF Model)
When the account setting capability allows, the user account privileges may be modified by issuing a set() operation on the ActivityQualifiers property of the CIM_Privilege class instance associated with the CIM_Account class instance. The set() operation requires an instance reference. The instance reference may be retrieved by adding EnumerateEPR mode to enumerate or get of the class.

The profile defines this property as string array containing all the privileges to be granted for the account. Setting the list of privileges is a complete over-write of the previous setting. This restriction is a limitation where the protocol does not define how to set a particular index in the list. The new list will replace the previous list in its entirety.

Profiles:

http://www.dmtf.org/sites/default/files/standards/documents/DSP1034_1.0.1.pdf
http://www.dmtf.org/sites/default/files/standards/documents/DSP1039_1.0.0.pdf
Here is an example list of available privileges from an instance of the class CIM_RoleBasedManagementCapabilities:

```
DCIM_LocalRoleBasedManagementCapabilities
  ActivitiesSupported = 7, 7, 7, 7, 7, 7, 7, 7, 7
  ActivityQualifiersSupported = Login to DRAC, Configure DRAC, Configure Users, Clear Logs, Execute Server Control Commands, Access Console Redirection, Access Virtual Media, Test Alerts, Execute Diagnostic Commands
  ElementName = Local Role Based Management Capabilities
  InstanceID = DCIM:LocalRoleBasedManagementCapabilities
  QualifierFormatsSupported = 9, 9, 9, 9, 9, 9, 9, 9, 9
  SharedPrivilegeSupported = false
  SupportedMethods = 8
```

The privilege property ActivityQualifiers is an array of type string. To set more than one privilege, you need to provide the same key name more than once. The tool does not allow duplicate keys to be entered through the command line. Instead, you need to perform two operations.

1. Get an instance of the CIM_Privilege class of interest.
2. Using the class instance, replace the property ActivityQualifiers with the new values.
3. Use the new instance XML as input to the set operation.

To determine if the new password has been successfully set, try logging in with the new password. Ensure the account is enabled.

6 Firmware Inventory

6.1 Software Inventory Profile Specification
The Dell Common Information Model (CIM) class extensions for supporting remote firmware inventory are defined in the Dell OS Software Update\(^\text{2}\) and related MOFs\(^\text{3}\). The diagrams representing the classes that are implemented by the Lifecycle Controller firmware can be found in Dell Software Inventory Profile.

6.2 Remote Inventory Method Invocation – Get Software Inventory
The SoftwareIdentity class contains information for the BIOS and component firmware installed on the target system as well as available firmware images cached in the Lifecycle Controller. The enumeration of the SoftwareIdentity class returns a list of SoftwareIdentity objects with properties such as firmware type and version.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

EXAMPLE:
When this method is executed, a list of software identity objects will be returned, including installed and available firmware. Below is a snippet of the output.

**OUTPUT:**

```
DCIM_SoftwareIdentity
   BuildNumber = 4846
   Classifications = 10
   ComponentID = 28897
   ComponentType = APAC
   DeviceID = null
   ElementName = Dell Lifecycle Controller 2, 1.0.0.4846, X79
   FQDD = USC.Embedded.1:LC.Embedded.1
   IdentityInfoType = OrgID:ComponentType:ComponentID
   IdentityInfoValue = DCIM:firmware:28897
   InstallationDate = 2012-01-15T22:22:32Z
   InstanceID = DCIM:INSTALLED#802__USC.Embedded.1:LC.Embedded.1
   IsEntity = true
   MajorVersion = 1
   MinorVersion = 0
   RevisionNumber = 0
   RevisionString = null
   Status = Installed
   SubDeviceID = null
   SubVendorID = null
   Updateable = true
   VendorID = null
   VersionString = 1.0.0.4846
   impactsTPMmeasurements = false
```

The key properties in the above output include the following:

**InstanceID:** Normally identifies the firmware on a particular type of device. The substring right after DCIM: is the status of a payload or firmware on the system. This can be **installed** or **available**.

**ComponentID:** Uniquely identifies a unique type of device such as BIOS, NIC, Storage and Lifecycle controller firmware.
**InstallationDate**: The date when the payload was installed to the system. If the system time was not set when the firmware installation took place the install date will be 1970-01-01. Factory installed firmware will have the 1970-01-01 date.

**VersionString**: Shows the version of the firmware represented.

# 7 Firmware Update

## 7.1 Software Update Profile Specification

The Dell Common Information Model (CIM) class extensions for supporting BIOS, component firmware, and embedded software update are defined in the Dell Software Update Profile and related MOF files. The diagrams representing the classes that are implemented by the Lifecycle Controller firmware can be found in Dell Software Update Profile as well.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

## 7.2 "Rollback" Firmware

The `InstallFromSoftwareIdentity()` method is used for installation of a previous version of a component firmware that is available on the Lifecycle Controller (i.e. “rollback” of component firmware). The general “Rollback” firmware steps are performed in several stages as described in the next sections. Meanwhile, the steps are demonstrated in examples in Section 7.3 and Section 7.4.

### 7.2.1 Request “Rollback” Image

The first stage is a request to initiate and download the rollback image from the Lifecycle Controller by invoking the `InstallFromSoftwareIdentity()` method.

### 7.2.2 Create Reboot Job

The second stage is to create a reboot job as shown in Section 7.8.

### 7.2.3 Schedule Update Jobs

The third stage is to invoke the `SetupJobQueue()` method as shown in Section 10.2.1. Use the `jobID(JID)` from `InstallFromSoftwareIdentity()` and `rebootID(RID)` from the reboot job. The reboot may take several minutes as the UEFI performs the desired operation.

### 7.2.4 Monitor Update Jobs

The output of getting the job status during various steps, Section 10.2.3, is shown below.

Initial job status after invoking `InstallFromSoftwareIdentity`
DCIM_LifecycleJob
InstanceID = JID_001276741956
JobStartTime = TIME_NA
JobStatus = Downloaded
JobUntilTime = TIME_NA
Message = Package successfully downloaded.
MessageArguments = null
MessageID = RED002
Name = Rollback:DCIM:AVAILABLE:NONPCI:159:2.1.4

Job status after invoking SetupJobQueue

DCIM_LifecycleJob
InstanceID = JID_001276741956
JobStartTime = 00000101000000
JobStatus = Scheduled
JobUntilTime = 20100730121500
Message = Task successfully scheduled
MessageArguments = null
MessageID = JCP001
Name = Rollback:DCIM:AVAILABLE:NONPCI:159:2.1.4

Job status following reboot / install of operation

DCIM_LifecycleJob
InstanceID = JID_001276741956
JobStartTime = 00000101000000
JobStatus = Completed
JobUntilTime = 20100730121500
Message = Job finished successfully
MessageArguments = null
MessageID = USC1
Name = Rollback:DCIM:AVAILABLE:NONPCI:159:2.1.4

7.3 BIOS Firmware “Rollback”
The InstallFromSoftwareIdentity() method is used for installation of a previous version of a component firmware that is available on the Lifecycle Controller (i.e. “rollback” of component firmware).

All steps to complete a rollback successfully are listed below.

Invoke InstallFromSoftwareIdentity() with the following parameters and syntax:

[InstanceID]: This is the instanceID of the SoftwareIdentity that is to be used to rollback the firmware to a previous version. The InstanceID can have value such as:

DCIM:AVAILABLE:NONPCI:159:2.1.4

• It is available firmware on a NONPCI device.
- This refers BIOS version 2.1.4

**EXAMPLE:**

```plaintext
winrm i InstallFromSoftwareIdentity cimv2/root/dcim/
DCIM_SoftwareInstallationService
?CreationClassName=DCIM_SoftwareInstallationService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=IDRAC:ID
+Name=SoftwareUpdate -file:RollInputBIOS.xml
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -auth:basic -encoding:utf-8
```

The rollback input file, `RollInputBIOS.xml`, is shown below:

```xml
    <a:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:Address>
    <a:ReferenceParameters>
      <w:SelectorSet>
        <w:Selector Name="InstanceID">[InstanceID]</w:Selector>
      </w:SelectorSet>
    </a:ReferenceParameters>
  </p:Target>
</p:InstallFromSoftwareIdentity_INPUT>

**OUTPUT:**

When this method is executed, a **jobid** or an error message is returned.

```plaintext
InstallFromSoftwareIdentity_OUTPUT
  Job
    Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
  ReferenceParameters
    ResourceURI = http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_SoftUpdateConcreteJob
    SelectorSet
      Selector: InstanceID = JID_001276741956,
      __cimnamespace = root/dcim
    ReturnValue = null
```

### 7.4 NIC Firmware “Rollback”

The `InstallFromSoftwareIdentity()` method is used for installation of a previous version of a component firmware that is available on the Lifecycle Controller (i.e. “rollback” of component firmware).
Invoke *InstallFromSoftwareIdentity* with the following parameters and syntax:

**[InstanceId]:** This is the instanceId of the SoftwareIdentity that is to be used to rollback the firmware to a previous version. The InstanceId can have value such as:


- It refers to a *previous* firmware on a PCI device.
- VID (Vendor ID)= 14E4
- DID (Device ID) = 1639
- SSID (Subsystem ID) = 0237
- SVID (Subvendor ID) = 1028
- This refers to a Broadcom NetXtreme II BCM5709 network adaptor.

**EXAMPLE:**

```plaintext
winrm i InstallFromSoftwareIdentity cimv2/root/dcim/DCIM_SoftwareInstallationService
?CreationClassName=DCIM_SoftwareInstallationService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=IDRAC:ID
+Name=SoftwareUpdate file: RollInputNIC.xml
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -auth:basic -encoding:utf-8
```

The rollback input file, *RollInputNIC.xml*, is shown below:

```xml
            xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
    <a:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:Address>
    <a:ReferenceParameters>
      <w:SelectorSet>
        <w:Selector Name="InstanceId">[InstanceId]</w:Selector>
      </w:SelectorSet>
    </a:ReferenceParameters>
  </p:Target>
</p:InstallFromSoftwareIdentity_INPUT>
```

**OUTPUT:**

When this method is executed, a *jobid* or an error message is returned.
ReferenceParameters
  SelectorSet
    Selector: InstanceID = JID_001265811668,
    __cimnamespace = root/dcim
ReturnValue = null

Entering an invalid *instanceID* may yield the following error message:

```
InstallFromSoftwareIdentity_OUTPUT
  Message = Invalid InstanceID
  MessageID = SUP024
  ReturnValue = null
```

### 7.5 Update from Network Source

A Firmware update can be achieved by invoking the `InstallFromURI()` method in the class `DCIM_SoftwareInstallationService`. Firmware update is performed in several stages as described in the next sections. The steps are demonstrated in examples in Section 7.6 and Section 7.7.

**Note:** When using WSMAN command to initiate update jobs, make sure to wait for two seconds before submitting a second job in order to avoiding racing conditions.

#### 7.5.1 Request Update Download

The first stage is a request to initiate and download the update image from a source defined by the user by invoking the `InstallFromURI()` method.

#### 7.5.2 Monitor Download Status

Downloading the update package may take several minutes. The second stage is to monitor the download. The download status may be monitored by enumerating or getting the instance of the corresponding job.

#### 7.5.3 Reboot to Perform Update

Once downloaded, the request needs to be scheduled. The third stage is to schedule the update. To schedule the update, use the `SetupJobQueue()` method of the class `DCIM_JobService` in Section 10.2.1.

#### 7.5.4 Wait for Job Completion

The fourth stage is to wait for the job to be completed, which may take several minutes. The job status can be monitored as shown in Section 10.2.3.
7.5.5 Delete Job
The fifth and final stage is to delete the completed job from the job store. Deleting the job queue is shown in Section 10.2.2.

7.6 Update NICs from HTTP, CIFS Share, NFS share, TFTP, or FTP
The InstallFromURI() method takes the following input and downloads the Dell Update Package to the Lifecycle Controller in the target system. The method returns a jobid for an instance of DCIM_SoftwareUpdateJob that can be scheduled to execute or queried for status at a later time. The following is the example of the method for updating a NIC firmware.

Invoke InstallFromURI() with the following parameters and syntax:

[URI-IP-ADDRESS]: This is the IP address of the location for Dell Update Package. The Dell Update Package will need to be the Windows type update package. The file share can be HTTP, CIFS, NFS, TFTP, or FTP type as shown below:

HTTP Format:
http://[IP ADDRESS]/[PATH TO FILE.exe]

CIFS Format:
cifs://WORKGROUP_NAME\[USERNAME]:[PASSWORD]@[URI-IP-ADDRESS]/[FILE.exe];mountpoint=[DIRECTORYNAME]

TFTP or FTP Format:
tftp://[IP ADDRESS]/[PATH TO FILE.exe]
ftp://[IP ADDRESS]/[PATH TO FILE.exe]

[InstanceID]: The instanceID is the SoftwareIdentify instanceID that represents the firmware that is to be updated. This instanceID can be retrieved as described in Section 6.2. For example, the instanceID can be:


- It is installed firmware on a PCI device.
- VID (Vendor ID) = 14E4
- DID (Device ID) = 1636
- SSID (Subsystem ID) = 0237
- SVID (Subvendor ID) = 1028
- This refers to a Broadcom NetXtreme II BCM5709 network adaptor².
EXAMPLE:

```
winrm invoke InstallFromURI cimv2/root/dcim/DCIM_SoftwareInstallationService
?CreationClassName=DCIM_SoftwareInstallationService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=IDRAC:ID+Name=SoftwareUpdate
-file:UpdateInputNIC.xml
-u:[UserName] -p:[Password] -r:https://[IPADDRESS]/wsman:443
-SkipCNCheck -auth:basic -encoding:utf8
```

The above command takes in an input file named `UpdateInputNIC.xml` to supply input parameters required for the `InstallFromURI()` method.

The syntax for `UpdateInputNIC.xml` is:

```xml
  <p:URI>http://[URI-IP-ADDRESS]/[PATH-TO-EXE]/[FILE.exe]</p:URI>
    <a:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:Address>
    <a:ReferenceParameters>
      <w:SelectorSet>
        <w:Selector Name="InstanceID">[INSTANCEID]</w:Selector>
      </w:SelectorSet>
    </a:ReferenceParameters>
  </p:Target>
</p:InstallFromURI_INPUT>
```

In the above sample, the `[URI-IP-ADDRESS]` must be replaced with the actual value of the IP address of the server that stores update content, `[PATH-TO-EXE]` must be replaced with the applicable path to the executable, `[FILE.exe]` must be replaced with the executable name, and `[INSTANCEID]` should be replaced with the actual `InstanceID` of the device to be updated.

OUTPUT:

When this method is executed, a `jobid` or an error message is returned. This `jobid` can then be used for subsequent processing with job control provider in Section 10.

```
InstallFromURI_OUTPUT
  Job
    Address = http://schemas.xmlsoap.org/ws
    ReferenceParameters
      ResourceURI = http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_SoftwareUpdateConcreteJob
      SelectorSet
        Selector: InstanceID = JID_001265810325,
        __cimnamespace = root/dcim
```
ReturnValue = null

Missing XML parameters may yield the following error message:

InstallFromURI_OUTPUT
  Message = Insufficient Method Parameters
  MessageID = SUP001
  ReturnValue = null

7.7 Update BIOS from HTTP, CIFS Share, NFS share, TFTP, or FTP

The InstallFromURI() method takes the following input and downloads the Dell Update Package to the Lifecycle Controller in the target system. The method returns a jobid for an instance of DCIM_SoftwareUpdateJob that can be scheduled to execute or queried for status at a later time. The following is the example of the method for updating a BIOS firmware.

Invoke InstallFromURI() with the following parameters and syntax:

  [URI-IP-ADDRESS]: This is the IP address of the location for Dell Update Package. The Dell Update Package will need to be the Windows type update package. The file share can be HTTP, CIFS, NFS, TFTP, or FTP type as shown below:

  HTTP Format:
  http://[IP ADDRESS]/[PATH TO FILE.exe]

  CIFS Format:
  cifs://[USERNAME]:[PASSWORD]@[URI-IP-ADDRESS]/[FILE.exe];mountpoint=/[DIRECTORYNAME]

  TFTP or FTP Format:
  tftp://[IP ADDRESS]/[PATH TO FILE.exe]
  ftp://[IP ADDRESS]/[PATH TO FILE.exe]

  [InstanceId]: The instanceId is the SoftwareIdentify instanceId that represents the firmware that is to be updated. This instanceId can be retrieved as described in Section 6.2. For example, the instanceId can be:

  DCIM:AVAILABLE:NONPCI:159:2.1.4

  - It is available firmware on a NONPCI device.
  - This refers BIOS version 2.1.4

  EXAMPLE:
The above command takes in an input file named `UpdateInputBIOS.xml` to supply input parameters required for the `InstallFromURI()` method.

The syntax for `UpdateInputBIOS.xml` is:

```xml
  <p:URI>http://[URI-IP-ADDRESS]/[PATH-TO-EXE]/[FILE.exe]/</p:URI>
  <p:Target xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
    <a:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:Address>
    <a:ReferenceParameters>
      <w:SelectorSet>
        <w:Selector Name="InstanceID">[INSTANCEID]</w:Selector>
      </w:SelectorSet>
    </a:ReferenceParameters>
  </p:Target>
</p:InstallFromURI_INPUT>
```

In the above sample, the `[URI-IP-ADDRESS]` must be replaced with the actual value of the IP address of the server that stores update content, `[PATH-TO-EXE]` must be replaced with the applicable path to the executable, `[FILE.exe]` must be replaced with the executable name, and `[INSTANCEID]` should be replaced with the actual `InstanceID` of the device to be updated.

**OUTPUT:**

When this method is executed, a `jobid` or an error message is returned. This `jobid` can then be used for subsequent processing with job control provider in section 10.
7.8 CreateRebootJob()

The CreateRebootJob() method creates a reboot job that can be scheduled to reboot immediately or at a later time. When the reboot job is scheduled and then executed, via SetupJobQueue() (Section 10.2.1), the reboot will take several minutes depending on the system setup, including whether collecting system inventory (CSIOR) is enabled.

Invoke CreateRebootJob with the following parameters and syntax:

**RebootJobType**: There are three options for rebooting the system.

1 = PowerCycle

2 = Graceful Reboot without forced shutdown

3 = Graceful reboot with forced shutdown

**EXAMPLE**:  

winrm invoke CreateRebootJob cimv2/root/dcim/DCIM_SoftwareInstallationService
?CreationClassName=DCIM_SoftwareInstallationService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=IDRAC:ID+Name=SoftwareUpdate
-file:reboot.xml
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-SkipCNCheck -auth:basic -encoding:utf-8

The syntax for reboot.xml is:

```xml
  <p:RebootJobType>2</p:RebootJobType>
</p:CreateRebootJob_INPUT>
```

**OUTPUT**:  

This method will return a reboot *jobid* that can be set to reboot the system immediately or at a later time.

CreateRebootJob_OUTPUT
RebootJobID
  Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
  ResourceURI = http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_SoftUpdateConcreteJob
SelectorSet
  Selector: InstanceID = RID_001265648530, __cimnamespace = root/dcim
ReturnValue = null

The *jobid* in the above output is the *instanceID*:
8  Power State Management

8.1  Description of Base Server vs Power State Management Methods
The remote control of a server power state (On, Off) and methodology for cycling power is available through data models specified in both the DMTF Base Server Profile and the DMTF Power State Management Profile. The Base Server Profile offers the RequestStateChange() method on the instance of the CIM_ComputerSystem class representing the server platform. The Power State Management Profile offers the RequestPowerStateChange() method available on the instance of the PowerStateManagementService associated with the instance of CIM_ComputerSystem representing the server platform.

Base Server Profile:
http://www.dmtf.org/sites/default/files/standards/documents/DSP1004_1.0.1.pdf

Power State Management Profile:
http://www.dmtf.org/sites/default/files/standards/documents/DSP1027_2.0.0.pdf

8.2  Get Power State

8.2.1  Base Server Method
The power state of the system is reported by the EnabledState property of the DCIM_ComputerSystem class.

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/DCIM_ComputerSystem
-u:[USER] -p:[PASSWORD]
-rt://[IPADDRESS]/wsman -SkipCNcheck -SkipCCheck
-encoding:utf-8 -a:basic

OUTPUT:

DCIM_ComputerSystem
  CreationClassName = DCIM_ComputerSystem
  Dedicated = 0
  ElementName
  EnabledState = 2
  HealthState = 25
  IdentifyingDescriptions = CIM:GUID, CIM:Tag, DCIM:ServiceTag
  Name = srv:system
  OperationalStatus = 6
OtherIdentifyingInfo = 4c4c4544-0036-3510-8034-b7c04f333231, mainsystemchassis, 7654321
PrimaryStatus = 3
RequestedState = 0

8.2.2 Power State Management Method
The power state of the system is also reported by the PowerState property of the DCIM_CSAssociatedPowerManagementService class.

Power State Management Profile:
http://www.dmtf.org/sites/default/files/standards/documents/DSP1027_2.0.0.pdf

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/DCIM_CSAssociatedPowerManagementService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic

OUTPUT:

PowerState:

  2 (On): System is fully on
  13 (Off): System is powered off

DCIM_CSAssociatedPowerManagementService
  PowerOnTime = null
  PowerState = 2
  RequestedPowerState = 0
  ServiceProvided
    EndpointReference
      Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    SelectorSet
      Selector: SystemCreationClassName = DCIM_SPComputerSystem, CreationClassName = DCIM_CSPowerManagementService, SystemName = systemmc, Name = pwrmgtsvc:1, __cimnamespace = root/dcim
      UserOfService
        EndpointReference
          Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
          ReferenceParameters
8.3 Get Power Control Capabilities

8.3.1 Base Server Method
The power control capabilities are reported by the RequestedStatesSupported property of the CIM_EnabledLogicalElementCapabilities class associated with the main system CIM_ComputerSystem class.

Base Server Profile:

http://www.dmtf.org/sites/default/files/standards/documents/DSP1004_1.0.1.pdf

In “Part A” enumerate the CIM_ElementCapabilities class and search for the DCIM_CSElementCapabilities reference. Use the resulting InstanceID in “Part B” to obtain the RequestedStatesSupported property.

EXAMPLE (Part A):

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ElementCapabilities
-u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

OUTPUT (Part A):

```
DCIM_CSElementCapabilities
Capabilities
  Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
  ReferenceParameters
  SelectorSet
    Selector: InstanceID = DCIM:ComputerCap:1, __cimnamespace = root/dcim
    Characteristics = null
    ManagedElement
      Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      ReferenceParameters
      SelectorSet
        Selector: Name = srv:system, CreationClassName = DCIM_ComputerSystem, __cimnamespace = root/dcim
```

47
EXAMPLE (Part B):

```
winrm g http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_CSEnabledLogicalElementCapabilities ?__cimnamespace=root/dcim
+InstanceID=DCIM:ComputerCap:1
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

OUTPUT (Part B):

```
RequestedStatesSupported:

  2: Enabled
  3: Disabled
  11: Reset
```

```
DCIM_CSEnabledLogicalElementCapabilities
  Caption = null
  Description = null
  ElementName = Computer System Capabilities
  ElementNameEditSupported = false
  ElementNameMask = null
  InstanceID = DCIM:ComputerCap:1
  MaxElementNameLen = null
  RequestedStatesSupported = 2, 3, 11
  StateAwareness = null
```

### 8.3.2 Power State Management Method

The power control capabilities are also reported by the `PowerStatesSupported` property of the `CIM_PowerManagementCapabilities` (PMC) class associated with the `CIM_PowerManagementService` (PMS) class. Getting the instance of PMC is a two step process. First, enumerate the instance of PMS with EPR. Second, enumerate the associated PMC class. When there is only one instance of PMC class as in the case of iDRAC, the first step may be skipped and the PMC class may be enumerated directly.

Power State Management Profile:

```
http://www.dmtf.org/sites/default/files/standards/documents/DSP1027_2.0.0.pdf
```

EXAMPLE (iDRAC case):

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_PowerManagementCapabilities?__cimnamespace=root/dcim
-u:[USER] -p:[PASSWORD]
```
When the PowerStatesSupported property contains the value in the “PowerStatesSupported Value” column, the PowerChangeCapabilities property shall contain the value specified in the “PowerChangeCapabilities Value” column.

<table>
<thead>
<tr>
<th>PowerStatesSupported Value</th>
<th>PowerChangeCapabilities Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (Power On)</td>
<td></td>
</tr>
<tr>
<td>3 (Sleep - Light)</td>
<td></td>
</tr>
<tr>
<td>4 (Sleep - Deep)</td>
<td>3 (Power State Settable)</td>
</tr>
<tr>
<td>5 (Power Cycle (Off Soft))</td>
<td>4 (Power Cycling Supported)</td>
</tr>
<tr>
<td>6 (Power Off - Hard)</td>
<td></td>
</tr>
<tr>
<td>7 (Hibernate)</td>
<td></td>
</tr>
<tr>
<td>8 (Power Off - Soft)</td>
<td></td>
</tr>
<tr>
<td>9 (Power Cycle (Off Hard))</td>
<td>6 (Off Hard Power Cycling Supported)</td>
</tr>
<tr>
<td>10 (Master Bus Reset)</td>
<td>7 (HW Reset Supported)</td>
</tr>
<tr>
<td>11 (Diagnostic Interrupt (NMI))</td>
<td>7 (HW Reset Supported)</td>
</tr>
<tr>
<td>12 (Power Off - Soft Graceful)</td>
<td>8 (Graceful Shutdown Supported)</td>
</tr>
<tr>
<td>13 (Power Off - Hard Graceful)</td>
<td>8 (Graceful Shutdown Supported)</td>
</tr>
<tr>
<td>14 (Master Bus Reset Graceful)</td>
<td>7 (HW Reset Supported) and 8 (Graceful Shutdown Supported)</td>
</tr>
<tr>
<td>15 (Power Cycle (Off - Soft Graceful))</td>
<td>4 (Power Cycling Supported) and 8 (Graceful Shutdown Supported)</td>
</tr>
<tr>
<td>16 (Power Cycle (Off - Hard Graceful))</td>
<td>6 (Off Hard Power Cycling Supported) and 8 (Graceful Shutdown Supported)</td>
</tr>
</tbody>
</table>

DCIM_CSPowerManagementCapabilities
Caption = null
Description = null
ElementName = Power Management Capabilities
InstanceID = DCIM:pwrmgtpcap1
OtherPowerCapabilitiesDescriptions = null
OtherPowerChangeCapabilities = null
PowerCapabilities = null
PowerChangeCapabilities = 3, 4, 8
PowerStatesSupported = 2, 5, 8, 11, 12

## 8.4 Power Control

### 8.4.1 Base Server Method
Changing the power state, such as cycling the power, is performed by invoking the RequestStateChange() method of the CIM_ComputerSystem class instance. For iDRAC, there is one
instance for the main system and another for iDRAC. Use the main system instance. The method requires you to specify the `RequestedState` argument. Refer to Section 8.3 to get the possible values for this argument.

**Base Server Profile:**
http://www.dmtf.org/sites/default/files/standards/documents/DSP1004_1.0.1.pdf

**EXAMPLE:**

```
winrm invoke RequestStateChange "http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_ComputerSystem
?CreationClassName=DCIM_ComputerSystem
+Name=srv:system"
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-SkipCNcheck
-SkipCAcheck -encoding:utf8 -a:basic @{'RequestedState="2"'}
```

**OUTPUT:**

```
RequestStateChange_OUTPUT
ReturnValue = 0
```

Return values of zero indicate success, while others indicate failure and may include a corresponding error message.

### 8.4.2 Power State Management Method

Changing the power state is performed by invoking the `RequestPowerStateChange()` method of the `DCIM_PowerManagementService` (PMS) class instance. It is a three step process shown below:

1) Enumerate the `DCIM_PowerManagementService` with EPR

2) Enumerate the `DCIM_ComputerSystem` class and search for the Host instance

3) Use the EPR on steps 1) and 2) to invoke RequestPowerStateChange()

**Power State Management Profile:**
http://www.dmtf.org/sites/default/files/standards/documents/DSP1027_2.0.0.pdf

**EXAMPLE:**

```
winrm invoke RequestPowerStateChange http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_CSPowerManagementService?__cimnamespace=root/dcim+SystemCreationClassName=DCIM_SPComputerSystem+SystemName=systemmc+CreationClassName=DCIM_CSPowerManagementService+Name=pwrmgtsvc:1
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck
-SkipCAcheck -encoding:utf8 -a:basic @{'PowerState="5"'}
```
9  Hardware Inventory

The Dell Common Information Model (CIM) class extensions for supporting remote hardware inventories are defined in the various Dell profiles and related MOFs\(^3\). The Hardware Inventory allows users to remote query the inventory of hardware.

Each of the hardware inventory classes return the attribute `LastSystemInventoryTime`, which is when the last time ‘collect system inventory on restart’ or CSIOR was run. See Section 12.1 for more details on CSIOR. It is an important attribute as it shows how recently the inventory was updated.

9.1  Power Supply Inventory

This section describes the implementation for the `DCIM_PowerSupplyView` class. The Dell Power Supply Profile describes platform’s power supply information. Each platform power supply is represented by an instance of `DCIM_PowerSupplyView` class.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library_Profile

Enumerate `DCIM_PowerSupplyView` with the following parameters and syntax:

**EXAMPLE:**

```
winrme cimv2/root/dcim/DCIM_PowerSupplyView
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]:wsman:443 -auth:basic
-encoding:utf-8 -SkipCACheck -SkipCNCheck
```

**OUTPUT:**

```
DCIM_PowerSupplyView
 DetailedState = Presence Detected
 FQDD = PSU.Slot.1
 FirmwareVersion = 00.01.31
 InputVoltage = 120
 InstanceID = PSU.Slot.1
 LastSystemInventoryTime = 20100331101859
 LastUpdateTime = 20100401130928
 Manufacturer = Dell
 Model = PWR SPLY,502W,RDNT
 PartNumber = 0MU791A00
 PrimaryStatus = 1
 RedundancyStatus = 2
 SerialNumber = CN732459700411
 TotalOutputPower = 502
 Type = 0
```

```
DCIM_PowerSupplyView
 DetailedState = Presence Detected
 FQDD = PSU.Slot.2
 FirmwareVersion = 00.01.31
```
InputVoltage = 118
InstanceID = PSU.Slot.2
LastSystemInventoryTime = 20100331101859
LastUpdateTime = 20100401130929
Manufacturer = Dell
Model = PWR SPLY, 502W, RDNT
PartNumber = 0MU791A00
PrimaryStatus = 1
RedundancyStatus = 2
SerialNumber = CN732459700446
TotalOutputPower = 502
Type = 0

9.2 Fan Inventory
This section describes the requirements and guidelines for implementing Dell Fan Profile. The Dell Fan Profile describes platform’s fans including the fan speed sensor information. Each platform fan is represented by an instance of DCIM_FanView class.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate DCIM_FanView with the following parameters and syntax:

EXAMPLE:

winrm e cimv2/root/dcim/DCIM_FanView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -auth:basic
-encoding:utf-8 -SkipCACheck -SkipCNCheck

OUTPUT:

DCIM_FanView
   ActiveCooling = true
   BaseUnits = 19
   CurrentReading = 4200
   FQDD = Fan.Embedded.1A
   InstanceID = Fan.Embedded.1A
   LastSystemInventoryTime = 20100331101859
   LastUpdateTime = 20100408115623
   PrimaryStatus = 1
   RateUnits = 4
   RedundancyStatus = 2
   UnitModifier = 0
   VariableSpeed = true

DCIM_FanView
   ActiveCooling = true
   BaseUnits = 19
   CurrentReading = 4440
   FQDD = Fan.Embedded.2A
   InstanceID = Fan.Embedded.2A
9.3 Memory Inventory
This section describes the implementation for the DCIM_MemoryView class. The Dell Memory Profile describes platform’s physical memory. Each DIMM’s information is represented by an instance of DCIM_MemoryView class.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate DCIM_MemoryView with the following parameters and syntax:

EXAMPLE:

winrm e cimv2/root/dcim/DCIM_MemoryView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -auth:basic
-encoding:utf-8 -SkipCACheck -SkipCNCheck

OUTPUT:

DCIM_MemoryView
  BankLabel = B
  CurrentOperatingSpeed = 1067
  FQDD = DIMM.Socket.B1
  InstanceID = DIMM.Socket.B1
  LastSystemInventoryTime = 20100331101859
  LastUpdateTime = 20100325134947
  ManufactureDate = Mon Jun 29 12:00:00 2009 UTC
  Manufacturer = Samsung
  MemoryType = 24
  Model = DDR3 DIMM
  PartNumber = M391B2873DZ1-CH9
  PrimaryStatus = 1
  Rank = 1
  SerialNumber = 85C6DF30
  Size = 1024
  Speed = 1333

DCIM_MemoryView
  BankLabel = A
  CurrentOperatingSpeed = 1067
FQDD = DIMM.Socket.A3
InstanceID = DIMM.Socket.A3
LastSystemInventoryTime = 20100331101859
LastUpdateTime = 20100325134947
ManufactureDate = Mon Jun 29 12:00:00 2009 UTC
Manufacturer = Samsung
MemoryType = 24
Model = DDR3 DIMM
PartNumber = M391B2873DZ1-CH9
PrimaryStatus = 1
Rank = 1
SerialNumber = 85C6DE0A
Size = 1024
Speed = 1333

9.4 CPU Inventory
This section describes the implementation for the DCIM_CPUView class. The Dell CPU Profile describes platform’s CPUs. Each CPU’s information is represented by an instance of DCIM_CPUView class.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate DCIM_CPUView with the following parameters and syntax:

```
EXAMPLE:

winrm e cimv2/root/dcim/DCIM_CPUView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -auth:basic
-encoding:utf-8 -SkipCACheck -SkipCNCheck
```

OUTPUT:

```
DCIM_CPUView
  CPUFamily = B3
  CPUStatus = 1
  Cache1Associativity = 7
  Cache1ErrorMethodology = 5
  Cache1Level = 0
  Cache1PrimaryStatus = 1
  Cache1SRAMType = 2
  Cache1Size = 256
  Cache1Type = 4
  Cache1WritePolicy = 0
  Cache2Associativity = 7
  Cache2ErrorMethodology = 5
  Cache2Level = 1
  Cache2PrimaryStatus = 1
  Cache2SRAMType = 2
```
Cache2Size = 2048
Cache2Type = 5
Cache2WritePolicy = 0
Cache3Associativity = 14
Cache3ErrorMethodology = 5
Cache3Level = 2
Cache3PrimaryStatus = 1
Cache3SRAMType = 2
Cache3Size = 20480
Cache3Type = 5
Cache3WritePolicy = 0
Characteristics = 4
CurrentClockSpeed = 2266
ExternalBusClockSpeed = 5860
FQDD = CPU.Socket.2
InstanceID = CPU.Socket.2
LastSystemInventoryTime = 20100331101859
LastUpdateTime = 20100325134947
Manufacturer = Intel
MaxClockSpeed = 3600
Model = Intel(R) Xeon(R) CPU E5520 @ 2.27GHz
NumberOfEnabledCores = 4
NumberOfEnabledThreads = 8
NumberOfProcessorCores = 4
PrimaryStatus = 1
Voltage = 1.20

DCIM_CPUView
  CPUFamily = B3
  CPUStatus = 1
  Cache1Associativity = 7
  Cache1ErrorMethodology = 5


9.5 iDRAC Card Inventory
This section describes the implementation for the DCIM_iDRACCardView class. The Dell iDrac Profile describes the platform’s iDrac remote access card. Each remote access card’s information is represented by an instance of DCIM_iDRACCardView class.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate DCIM_iDRACCardView with the following parameters and syntax:

EXAMPLE:

winrm e cimv2/root/dcim/DCIM_iDRACCardView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -auth:basic
OUTPUT:

DCIM_iDRACCardView
  FQDD = iDRAC.Embedded.1
  FirmwareVersion = 1.00.00
  GUID = 314b544f-c0b5-5180-5210-00484c4c54
  IPMIVersion = 2.0
  InstanceID = iDRAC.Embedded.1-1#IDRACinfo
  LANEnabledState = 1
  LastSystemInventoryTime = 20100331101859
  LastUpdateTime = 19700101000000
  Model = Enterprise
  PermanentMACAddress = 0:21:9b:92:70:5f
  ProductDescription = This system component provides a complete set of remote management functions for Dell PowerEdge server
  SOLEnabledState = 1
  URLString = https://192.35.10.1:443

9.6 PCI Device Inventory
This section describes the implementation for the DCIM_PCIDeviceView class. The Dell PCI Profile describes platform’s PCI devices. Each PCI device’s information is represented by an instance of DCIM_PCIDeviceView class.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate DCIM_PCIDeviceView with the following parameters and syntax:

EXAMPLE:

winrm e cimv2/root/dcim/DCIM_PCIDeviceView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -auth:basic
-encoding:utf-8 -SkipCACheck -SkipCNCheck

OUTPUT:

DCIM_PCIDeviceView
  BusNumber = 0
  DataBusWidth = 0002
  Description = 82801I (ICH9 Family) USB UHCI Controller #4
  DeviceNumber = 26
  FQDD = USBUHCI.Embedded.4-1
  FunctionNumber = 0
  InstanceID = USBUHCI.Embedded.4-1
  LastSystemInventoryTime = 20100331101859
  LastUpdateTime = 20100325134947
  Manufacturer = Intel Corporation
  PCIDeviceID = 2937
  PCISubDeviceID = 0236
  PCISubVendorID = 1028
PCIVendorID = 8086
SlotLength = 0002
SlotType = 0002

DCIM_PCIDeviceView
BusNumber = 0
DataBusWidth = 0002
Description = 5520/5500/X58 I/O Hub PCI Express Root Port 3
DeviceNumber = 3
FQDD = P2PBridge.Embedded.4-1
FunctionNumber = 0
InstanceID = P2PBridge.Embedded.4-1
LastSystemInventoryTime = 20100331101859
LastUpdateTime = 20100325134947
Manufacturer = Intel Corporation
PCIDeviceID = 340A
PCISubDeviceID = 0000
PCISubVendorID = 0000
PCIVendorID = 8086
SlotLength = 0002
SlotType = 0002

DCIM_PCIDeviceView
.
.
.

9.7 Video Inventory
This section describes the implementation for the DCIM_VideoView class. The Dell Video Profile describes platform’s videos. Each video controller’s information is represented by an instance of DCIM_VideoView class.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate DCIM_VideoView with the following parameters and syntax:

EXAMPLE:

winrm e cimv2/root/dcim/DCIM_VideoView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -auth:basic
-encoding:utf-8 -SkipCACheck -SkipCNCheck

OUTPUT:

DCIM_VideoView
BusNumber = 6
DataBusWidth = 0002
Description = PowerEdge R610 MGA G200eW WPCM450
DeviceNumber = 3
FQDD = Video.Embedded.1-1
FunctionNumber = 0
InstanceID = Video.Embedded.1-1
LastSystemInventoryTime = 20100331101859
LastUpdateTime = 20100325134947
Manufacturer = Matrox Graphics, Inc.
PCIDeviceID = 0532
PCISubDeviceID = 0236
PCISubVendorID = 1028
PCIVendorID = 102B
SlotLength = 0002
SlotType = 0002

9.8 VFlash SD Card Inventory

Each SD card partition is represented by an instance of DCIM_VFlashView that is used to represent the physical attributes of the virtual flash media, such as total size, available size, category etc. on which the partitions will reside. See Section 13 for more information.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate the DCIM_VFlashView with the following parameters and syntax:

EXAMPLE:

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_VFlashView
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

OUTPUT:

```
DCIM_VFlashView
  AvailableSize = 970
  Capacity = 976
  ComponentName = vFlash SD Card
  FQDD = Disk.vFlashCard.1
  HealthStatus = OK
  InitializedState = Initialized
  InstanceID = Disk.vFlashCard.1
  LastSystemInventoryTime = 20100408123517
  LastUpdateTime = 20100408123517
  Licensed = true
  VFlashEnabledState = true
  WriteProtected = false
```

9.9 NIC Inventory & Configuration
The NIC Profile describes NIC controller’s representation and configuration. The profile also describes the relationship of the NIC classes to the DMTF/Dell profile version information. See Section 15 for more information, including inventories for NICString, NICInteger, and NICE Enumeration.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate NICView with the following parameters and syntax:

EXAMPLE:

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAncheck
-encoding:utf-8 -a:basic
```

OUTPUT:

```
DCIM_NICView
    AutoNegotiation = 0
    BusNumber = 1
    ControllerBIOSVersion = 1.3
    CurrentMACAddress = 00:21:9B:92:70:57
    DataBusWidth = 0002
    DeviceNumber = 0
    EFIVersion = null
    FCoEOffloadMode = 3
    FCoEWWNN = null
    FQDD = NIC.Embedded.1-1
    FamilyVersion = null
    FunctionNumber = 0
    InstanceID = NIC.Embedded.1-1
    LastSystemInventoryTime = 20100413135024
    LastUpdateTime = 20100413134727
    LinkDuplex = 0
    LinkSpeed = 0
    MaxBandwidth = 0
    MediaType = 4
    MinBandwidth = 0
    NicMode = 3
    PCIDeviceID = 1639
    PCISubDeviceID = 236
    PCISubVendorID = 1028
    PCIVendorID = 14E4
    PermanentFCOEMACAddress
    PermanentMACAddress = 00:21:9B:92:70:57
    PermanentiSCSIMACAddress = 00:21:9B:92:70:58
    ProductName = Broadcom NetXtreme Gigabit Ethernet - 00:21:9B:92:70:57
    ReceiveFlowControl = 0
    SlotLength = 0002
    SlotType = 0002
    TransmitFlowControl = 0
```
9.10 RAID Inventory & Configuration
The RAID profile extends the management capabilities of referencing profiles by adding the capability to represent the configuration of RAID storage. The RAID storage is modeled as collections of attributes where there are collections for the storage adaptors, physical disks, logical disks, end enclosures and parent-child relationships between the collections. Additionally, there is a configuration service that
contains all the methods used to configure the RAID storage. See Section 16 for more information, including inventories for PhysicalDiskView, VirtualDiskView, and EnclosureView.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate ControllerView with the following parameters and syntax:

EXAMPLE:

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_ControllerView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic
```

OUTPUT:

```
DCIM_ControllerView
  Bus = 3
  CacheSizeInMB = 1024
  CachecadeCapability = 1
  ControllerFirmwareVersion = 20.10.1-0049
  Device = 0
  DeviceCardDataBusWidth = 1
  DeviceCardManufacturer = DELL
  DeviceCardSlotLength = 3
  DeviceCardSlotType = PCI Express x8
  DriverVersion = null
  EncryptionCapability = 0
  EncryptionMode = 0
  FQDD = RAID.Integrated.1-1
  Function = 0
  InstanceID = RAID.Integrated.1-1
  LastSystemInventoryTime = 20100331101859
  LastUpdateTime = 20100330124133
  PCIDeviceID = 73
  PCISlot = 1
  PCISubDeviceID = 1F51
  PCISubVendorID = 1028
  PIVendorID = 1000
  PatrolReadState = 1
  PrimaryStatus = 0
  ProductName = PERC H310 Mini
  RollupStatus = 0
  SASAddress = 50026B902A8B6E00
  SecurityStatus = 0
  SlicedVDCapability = 1
```
9.11 BIOS Inventory & Configuration

The BIOS Management Profile extends the management capabilities of referencing profiles by adding the capability to represent and configure BIOS attributes, such as a Network Controller or IDE Controller. The individual BIOS attribute’s relationship with a respective device is also described. Additionally, the profile’s registration for the schema implementation version information is described. See Section 17 for more information, including inventories for BIOSString, and BIOSInteger.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate BIOSEnumeration with the following parameters and syntax:

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BIOSEnumeration
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic

OUTPUT:

DCIM_BIOSEnumeration
    AttributeDisplayName = Memory Operating Voltage
    AttributeName = MemVolt
    CurrentValue = AutoVolt
    Dependency = <Dep><AttrLev Op="OR"><ROIf Op="NOT"
Name="SysProfile">Custom</ROIf></AttrLev><ValLev Val="AutoVolt" Op="OR"><Forcelf
Name="SysProfile">PerfPerWattOptimizedDapc</Forcelf><Forcelf
Name="SysProfile">PerfPerWattOptimizedOs</Forcelf><Forcelf
Name="SysProfile">PerfOptimized</Forcelf><Suplf
Name="SysProfile">DenseCfgOptimized</Suplf><ValLev Val="Volt15V" Op="OR"><Forcelf
Name="SysProfile">DenseCfgOptimized</Forcelf><Suplf
Name="SysProfile">PerfPerWattOptimizedDapc</Suplf><Suplf
Name="SysProfile">PerfPerWattOptimizedOs</Suplf><Suplf
Name="SysProfile">PerfOptimized</Suplf><Suplf
Name="SysProfile">PerfOptimized</Suplf><ValLev><Dep>
DisplayOrder = 1322
FQDD = BIOS.Setup.1-1
GroupName = System Profile Settings
GroupID = SysProfileSettings
InstanceID = BIOS.Setup.1-1:MemVolt
IsReadOnly = true
PendingValue = null
PossibleValues = AutoVolt, Volt15V
PossibleValuesDescription = Auto, 1.5V

DCIM_BIOSEnumeration
    AttributeDisplayName = Serial Debug Output
    AttributeName = SerialDbgOut
    CurrentValue = Disabled
    Dependency = null
DisplayOrder = 319  
FQDD = BIOS.Setup.1-1  
GroupDisplayName = Memory Settings  
GroupID = MemSettings  
InstanceID = BIOS.Setup.1-1:SerialDbgOut  
IsReadOnly = false  
PendingValue = null  
PossibleValues = Enabled, Disabled  
PossibleValuesDescription = Enabled, Disabled

DCIM_BIOSEnumeration  
AttributeDisplayName = Password Status  
AttributeName = PasswordStatus  
CurrentValue = Unlocked  
Dependency = null  
DisplayOrder = 1405  
FQDD = BIOS.Setup.1-1  
GroupDisplayName = System Security  
GroupID = SysSecurity  
InstanceID = BIOS.Setup.1-1:PasswordStatus  
IsReadOnly = false  
PendingValue = null  
PossibleValues = Unlocked, Locked  
PossibleValuesDescription = Unlocked, Locked

9.12 System Inventory (including CSIOR attribute)  
This section describes the implementation for the DCIM_SystemView class which is used to represent the higher level attributes of the system, such as asset tag, model, server manufacturer, etc.

Profile and Associated MOFs:  
http://www.delltechcenter.com/page/DCIM.Library.Profile

Enumerate SystemView with the following parameters and syntax:

EXAMPLE:

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/ DCIM_SystemView  
-u: [USER] -p: [PASSWORD]  
-r: https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck  
-encoding:utf-8 -a: basic
```

OUTPUT:

DCIM_SystemView  
AssetTag = Testtg  
BIOSReleaseDate = 09/12/2011  
BIOSVersionString = 0.3.22  
BaseBoardChassisSlot = NA  
BatteryRollupStatus = 1
BladeGeometry = 4
BoardPartNumber = 0N051FX02
BoardSerialNumber = CN13740920003M
CMCIP = null
CPLDVersion = 0.4.7
CPURollupStatus = 1
ChassisName = Main System Chassis
ChassisServiceTag = 7654321
ChassisSystemHeight = 2
ExpressServiceCode = 61387326761
FQDD = System.Embedded.1
FanRollupStatus = 3
HostName
InstanceID = System.Embedded.1
LastSystemInventoryTime = 20100331101859
LastUpdateTime = 20100325134947
LicensingRollupStatus = 1
LifecycleControllerVersion = 2.0.0
Manufacturer = Dell Inc.
MaxCPU.Sockets = 2
MaxDIMMSlots = 24
MaxPCIe.Slots = 3
MemoryOperationMode = OptimizerMode
Model = PowerEdge R620
PSRollupStatus = 1
PlatformGUID = 3548474f-c0d3-4680-3810-00374c4c4544
PopulatedCPU.Sockets = 1
PopulatedDIMMSlots = 1
PopulatedPCIe.Slots = 1
PowerCap = 0
PowerCapEnabledState = 3
PowerState = 2
PrimaryStatus = 3
RollupStatus = 3
ServerAllocation = null
ServiceTag = S78FGH5
StorageRollupStatus = 1
SysMemErrorMethodology = 6
SysMemFailOverState = NotInUse
SysMemLocation = 3
SysMemPrimaryStatus = 1
SysMemTotalSize = 2048
SystemGeneration = 12G Monolithic
SystemID = 1230
SystemRevision = 0
TempRollupStatus = 1
UUID = 4c4c4544-0037-3810-8046-d3c04f474835
VoltRollupStatus = 1
smbiosGUID = 44454c4c-3700-1038-8046-d3c04f474835
10  Job Control Management

10.1 Description of Job Management
The Dell Common Information Model (CIM) class extensions for supporting update and attribute configuration job control are defined in the Dell Job Control Profile\(^2\) and related MOF files\(^3\). The diagrams representing the classes that are implemented by the Lifecycle Controller firmware can be found in Dell Job Control Profile as well.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

10.2 Remote Job Control Examples

10.2.1 Setup Job Queue
The SetupJobQueue() method takes in an array of jobids and schedules them to run immediately or at a later time. The jobids are acquired via enumerating DCIM_LifecycleJob as described in Section 10.2.3. When there is a Reboot Job, in a job array that contains multiple jobs, the system will reboot the UEFI (Unified Extensible Firmware Interface) at the scheduled time.

Invoke SetupJobQueue() with the following parameters and syntax:

**JobArray:** The jobids are listed in the JobArray element. Multiple jobs are listed in the order of job execution sequence. If a system is to reboot at the scheduled start time, a reboot job will need to be added to the list. This reboot job has a prefix of RID_ for its jobid.

Note, scheduling a job that is already scheduled will result in an error message.

If there is no reboot job in the job array, the system will schedule the jobs for execution at the specified start time. The jobs will not be executed until the system is rebooted by something other than Lifecycle Controller. At the specified UntilTime, any jobs that have not been executed are failed with an error indicating that the job was not executed in the specified maintenance window. For some component updates such as Diagnostics, USC, and iDRAC firmware, a system reboot is not needed.

**EXAMPLE:**

winrm invoke SetupJobQueue cimv2/root/dcim/DCIM_JobService
?CreationClassName=DCIM_JobService
+Name=JobService+SystemName=Idrac
+SystemCreationClassName=DCIM_ComputerSystem
-file:SetupJobQueue.xml
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -auth:basic -encoding:utf-8

The syntax for SetupJobQueue.xml is:

```
<p:JobArray>JID_001249463339</p:JobArray>  
<p:JobArray>RID_001265817718</p:JobArray>
```
Here the JobArray element shows a list of Jobids that are to be scheduled to run. \textit{TIME\_NOW} is a special value that represents “running the tasks immediately”. The UntilTime value specifies the “maintenance windows”. Once a task is not run after passing UntilTime, it should not be run again.

Upon successfully invocation of the SetupJobQueue() method, the aforementioned times will be listed when enumerated in Section 10.2.3.

**OUTPUT:**

Returns 0 for success or non-zero for error with messageID and message description.

\begin{verbatim}
SetupJobQueue\_OUTPUT
  ReturnValue = null
\end{verbatim}

Entering an invalid jobid or XML syntax error can yield one of the following error messages:

\begin{verbatim}
SetupJobQueue\_OUTPUT
  Message = Job Cannot be Scheduled
  MessageID = SUP016
  ReturnValue = null
\end{verbatim}

\begin{verbatim}
SetupJobQueue\_OUTPUT
  Message = Duplicated/Invalid JOBID Entries
  MessageID = SUP023
  ReturnValue = null
\end{verbatim}

**10.2.2 Delete Job Queue**

The DeleteJobQueue() method takes in a jobID and then deletes it from the job store.

Note: When clearing all jobs and pending data using the keyword \textit{JID\_CLEARALL}, as shown in example 2 below, the remote services instrumentation is restarted to clear the cache \textbf{[LC 1.x ONLY]}. Users should allow two minutes for this process to complete.

Invoke DeleteJobQueue() with the following parameters and syntax:

\begin{verbatim}
 [JobID]: The jobID of a particular job instance to be deleted from a jobqueue
\end{verbatim}

**EXAMPLE 1:**

```
winrm invoke DeleteJobQueue cimv2/root/dcim/DCIM_JobService
?CreationClassName=DCIM_JobService
+Name=JobService+SystemName=Idrac
+SystemCreationClassName=DCIM_ComputerSystem
@{JobID="[jobID]" }
-u:[USER] -p:[PASSWORD]
```
The example below uses JID_CLEARALL for the jobID, which is a predefined value that represents “deleting all jobs in the jobstore”.

EXAMPLE 2:

```bash
winrm invoke DeleteJobQueue cimv2/root/dcim/DCIM_JobService
?CreationClassName=DCIM_JobService+Name=JobService
+SystemName=Idrac
+SystemCreationClassName=DCIM_ComputerSystem
@[JobID="JID_CLEARALL" ]
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-auth:basic -encoding:utf-8
```

OUTPUT:

Return 0 for success or non-zero for error with messageID and message description.

DeleteJobQueue_OUTPUT

Message = The specified job was deleted
MessageID = SUP020
ReturnValue = null

An XML syntax error could yield the following message:

Syntax Error: input must be of the form

```
{KEY="VALUE"[;KEY="VALUE"]}
```

10.2.3 List Jobs in Job Store
The instances of this class will enumerate jobs in the job store along with status information.

Invoke enumerate job status with the following parameters and syntax:

[JobID]: The JobID of a particular job instance to be queried

To get the status of one particular job, use the following:

EXAMPLE 1:

```bash
winrm get http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LifecycleJob
?InstanceID=[JobID]
-r:https://[IPADDRESS]/wsman:443
-u:[USERNAME] -p:[PASSWORD]
-a:basic -encoding:utf-8
```

To get the status of all jobs, use the following:

EXAMPLE 2:
winrm e cimv2/root/dcim/DCIM_LifecycleJob
-u:[USERNAME] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-auth:basic -encoding:utf-8

OUTPUT 1 & 2:

The method either returns a list of Concrete job objects or an error message. Once job instanceID are returned via these status queries, they can be used for job scheduling and setup. Several examples of job objects are shown below.

DCIM_LifecycleJob
InstanceID = JID_001275884806
JobStartTime
JobStatus = Completed
JobUntilTime
Message = Detach partition successful
MessageArguments = null
MessageID = VF038
Name = VFlashDetach:Partition3

DCIM_LifecycleJob
InstanceID = RID_001274051062
JobStartTime = 00000101000000
JobStatus = Reboot Completed
JobUntilTime = 20100730121500
Message
MessageArguments = null
MessageID
Name = Reboot3

DCIM_LifecycleJob
InstanceID = JID_001274140369
JobStartTime = 00000101000000
JobStatus = Completed
JobUntilTime = 20111111111111
Message = Job completed successfully
MessageArguments = null
MessageID = PR19
Name = ConfigRAID:RAID.Integrated.1-1

An error message similar to the following can occur if an invalid JobID is entered:

WSManFault

Message = The WinRM client cannot process the request. The destination computer returned an empty response to the request.

Error number: -2144108299 0x803380F5

The WinRM client cannot process the request. The destination computer returned an empty response to the request.
11 Operating System Deployment
The Dell Common Information Model (CIM) class extensions for supporting remote operating system (OS) deployment are defined in the Dell OS Deployment Profile\(^2\) and the \textit{DCIM\_OSDeploymentService MOF file}\(^3\). The diagrams representing the classes that are implemented by the Lifecycle Controller firmware can be found in Dell OS Deployment Profile as well.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

11.1 OS Deployment Profile Implementation Conformance
Use the following algorithm to test the instrumentation for OS Deployment Profile version conformance and to discover the implementation namespace:

1. Enumerate (namespace='root/interop', classname="CIM\_RegisteredProfile")
2. Filter the returned enumeration using property filter (RegisteredName="OS Deployment")
3. Result shall contain one instance of \textit{CIM\_RegisteredProfile} containing property RegisteredVersion="1.1.0"
4. Associators (objectpath= “instance returned from step 3”, AssociationClass = "CIM\_ElementConformsToProfile")
5. Result shall contain one instance of \textit{DCIM\_OSDeploymentService}

11.2 Checking OS Deployment Service Availability
Invoke \textit{enumerate} with the following syntax:

\textbf{EXAMPLE}:

```plaintext
winrm e cimv2/root/dcim/DCIM\_OSDeploymentService
-u:[USER]  -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-encoding:utf-8 -a:basic
```

\textbf{OUTPUT}:

```
DCIM\_OSDeploymentService
  CreationClassName = DCIM\_OSDeploymentService
  ElementName = Operating System Deployment Service
  Name = DCIM\:OSDeploymentService
  SystemCreationClassName = DCIM\:ComputerSystem
  SystemName = DCIM\:ComputerSystem
```

11.3 OS Deployment Method Invocation Examples
### 11.3.1 Get Driver Pack Information

The `GetDriverPackInfo()` method returns the embedded driver pack version and list of supported OSs for OS deployment that can be installed on the server using the embedded device drivers present in the Lifecycle Controller.

1. Follow the steps listed in Section 11.1 to test for profile conformance.

2. Invoke extrinsic method using the following parameters:
   a. object path = object path returned from Section 11.1 (profile conformance)
   b. Method name = “GetDriverPackInfo”

3. Invoke method returns the following output parameters:
   a. Version = String version
   b. SupportedOperatingSystems = String array of OS names

4. If the Job output parameter from Step 2 contains a non-null value, then both Version and OSList contain null values. The next call to `GetDriverPackInfo()` after the Job is completed will return non-null values for output parameters Version and OSList.

Invoke `GetDriverPackInfo()` with the following syntax:

**EXAMPLE**:  

```
winrm i GetDriverPackInfo
  cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem +SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman
-encoding:utf-8 -a:basic
```

**OUTPUT**:  

```
GetDriverPackInfo_OUTPUT
   OSList = Microsoft Windows Server 2008 with SP2
   , Microsoft Windows Server 2008, x64 with SP2
   , Microsoft Windows Server 2008 R2 with SP1
   , Microsoft Windows Small Business Server 2011
   , Red Hat Enterprise Linux 5 SP7 x86
   , Red Hat Enterprise Linux 5 SP7 x64
   , Red Hat Enterprise Linux 6 SP1 x64
   , SuSE Enterprise Linux 10 SP4 x64
   , SuSE Enterprise Linux 11 SP2 x64
   , VMware ESX 4.1 U2
   , VMware ESXi 4.1 U2 HDD
   , VMware ESXi 5.0 HDD
   , Citrix Xen Server 6.0 FP1 HDD
```

ReturnValue = 0
11.3.2 Unpack Selected Drivers and Attach to Host OS as USB Device

This method is used to unpack the drivers for the selected OS to a virtual storage partition, and to then attach this partition to the host OS as an emulated USB storage device.

1. Invoke extrinsic method using the following parameters section:
   a. object path = object path returned from Section 11.1 (profile conformance)
   b. Method name = “UnpackAndAttach”
   c. OSName = “” (Has to be a valid value from the list returned by GetDriverPackInfo)
   d. ExposureStartTime = “” (for this release the value is NULL)
   e. ExposureDuration = “” (a string formatted as an interval in CIM_DateTime format)
      This parameter denotes the interval of time after which the partition containing OS drivers with label OEMDRV is to be detached from the Host OS

2. Invoke method shall return the following output parameters:
   a. Job = object path to CIM_ConcreteJob (reports the status of unpack and attach)
   b. Enumerating this instance of CIM_ConcreteJob will show the status of the current operation.

Invoke UnpackAndAttach() with the following syntax:

**EXAMPLE:**

```
winrm i UnpackAndAttach cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u: [USER] -p: [PASSWORD]
-r: https://[IPADDRESS]:443
-encoding: utf-8 -a: basic
@{OSName="[OSName]";ExposureDuration="00000000002200.000000:000"}
```

Above example uses Microsoft Windows Server 2008 with SP2 for OSName.

**OUTPUT:**

UnpackAndAttach_OUTPUT

Job

Address = [http://schemas.xmlsoap.org/ws]
11.3.3 Detach Emulated USB Device Containing Drivers

This method is used to detach the USB device attached to the system by a previous invocation of the UnpackAndAttach() method.

Invoke DetachDrivers() with the following syntax:

**EXAMPLE:**

```bash
winrm i DetachDrivers cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-encoding:utf-8 -a:basic
```

**OUTPUT:**

The return will be 0 for success or an integer for error or job in execution. An error message containing a MessageID and Message similar to the following can occur if the system is waiting to finish a previously invoked method:

DetachDrivers_OUTPUT

- Message = Unable to retrieve Lifecycle Controller handle
- MessageID = OSD7
- ReturnValue = 2

11.3.4 Unpack Selected Drivers and Copy to Network Share

The UnpackAndShare() method is used to unpack the drivers for the selected OS and copy them to a specified network share; CIFS and NFS network share technologies are supported.

Note that the values for the CIFSUSER and CIFSPASSWORD must be alphanumeric characters, and must not contain special characters.

Invoke UnpackAndShare() with the following syntax:

- **[CIFS_IPADDRESS]**: This is the IP address of the file server.
[DRIVESHARE]: This is the directory path to the drivers.

[CIFS_USERNAME]: This is the username to the file share.

[CIFS_PASSWORD]: This is the password to the file share.

[OSName]: This example uses Windows Server® 2003 SP2.

[NFS_Password]: This is the corresponding password to the username containing the ISO

**EXAMPLE:**

```powershell
winrm i UnpackAndShare cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM:ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]:443/wsman
-encoding:utf-8 -a:basic
@{IPAddress="[CIFS_IPADDRESS]";ShareName="[DRIVESHARE]";ShareType="2";Username="[CIFS_USERNAME]";Password="[CIFS_PASSWORD]";OSName="Microsoft Windows Server 2008 with SP2"}
```

**OUTPUT:**

The return will be 0 for success or 1 if an error occurred in starting the processing the input parameters. The `MessageID` and `Message` output parameters will further contain method invocation information if an error occurred.

UnpackAndShare_OUTPUT

**Job**

- **Address** = [URL](http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous)
- **ReferenceParameters**
  - **ResourceURI** = [URL](http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_OSDConcreteJob)
- **SelectorSet**
  - **Selector**: InstanceID = DCIM_OSDConcreteJob:1,
  - **__cimnamespace** = root/dcim
- **ReturnValue** = 4096

A missing command line character, such as a “{“, could result in the following syntax error:

```
  Syntax Error: input must be of the form {KEY="VALUE"[;KEY="VALUE"]}
```

**11.3.5 Check Job Status**

The following methodology is used to determine the status of the jobs generated by the invocation of the `UnpackAndAttach()` and `UnpackAndShare()` methods. The methodology involves enumerating the `DCIM_OSDConcreteJob` instances, and checking the `JobStatus` property value.
When the jobs are complete, the `JobStatus` property value will be “Successful” if the job completed successfully or “Failed” if an error occurred while executing the request. If the job failed, the `Message` property on the returned `DCIM_OSDConcreteJob` instance will contain more detailed error information on the cause of the failure.

For the Lifecycle Controller version of the OS Deployment Profile there is only one instance of a job generated by various method invocations, and it will persist until the next method that generates a job is invoked. The job must complete before another method that generates a job can be called successfully. This is unchanged from the Lifecycle Controller 1.2 for OS Deployment.

Invoke `enumerate DCIM_OSDConcreteJob` instance with the following syntax:

**EXAMPLE:**

```text
winrm e cimv2/root/DCIM/DCIM_OSDConcreteJob
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-SkipCNCheck -encoding:utf-8 -a:basic
```

**OUTPUT:**

The enumeration will return the instances of `OSDConcreteJob` as shown:

```text
DCIM_OSDConcreteJob
  DeleteOnCompletion = false
  InstanceID = DCIM_OSDConcreteJob:1
  JobName = UnpackAndShare
  JobStatus = Failed
  Message = Installation not supported for the selected operating system
  MessageID = OSD10
  Name = UnpackAndShare
```

### 11.3.6 Boot to Network ISO

The `BootToNetworkISO()` method can be used to boot the target system to a bootable ISO image located on a CIFS or NFS share. The ISO image is attached to the host system as an emulated USB CD-ROM storage device. By default the ISO will be attached for around 18 hrs after which it will be detached automatically. An optional parameter `ExposureDuration` can be used to specify a time less than 18 hrs if the ISO needs to be detached sooner.

Invoke `BootToNetworkISO()` via NFS share with the following syntax:

- `[NFS_IPADDRESS]`: This is the IP address of the location of the ISO image.
- `[NFS/OSISO]`: This is the directory path to the ISO image.
- `[NFS_Username]`: This is the username to the IP address of the ISO image.
- `[NFS_Password]`: This is the corresponding password to the username containing the ISO image.
[OS.ISO]: This is to be replaced by the actual name of the ISO image.

**EXAMPLE:**

```bash
winrm i BootToNetworkISO
cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u: [USER] -p: [PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -SkipCNCheck
-encoding:utf-8 -a:basic
@{IPAddress="[NFS_IPaddress]";ShareName="/NFS/OSISO";ShareType="0";
Username="[NFS_Username]";Password="[NFS_Password]";
Workgroup="WORKGROUP";ImageName="[OS.ISO]"}
```

**OUTPUT:**

The return will be 0 for success or 1 if an error occurred in starting the processing the input parameters. The *MessageID* and *Message* output parameters will further contain method invocation information if an error occurred.

**BootToNetworkISO_OUTPUT**

```
Job
    Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
SelectorSet
    Selector: InstanceID = DCIM_OSDConcreteJob:1, __cimnamespace = root/dcim
    ReturnValue = 4096
```

The following error message is caused by a typo in the WinRM input. Careful attention must be paid to the input capitalization of the attributes.

**WSManFault**

```
Message = The WinRM client cannot process the request. The destination computer returned an empty response to the request.

Error number: -2144108299 0x803380F5
```

The WinRM client cannot process the request. The destination computer returned an empty response to the request.

**11.3.7 Detach Network ISO USB Device**
This method is used to detach the emulated USB device that had been attached by previously calling the `BootToNetworkISO()` method.

Invoke `DetachISOImage()` with the following syntax:

**EXAMPLE:**

```plaintext
winrm i DetachISOImage cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM_OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-encoding:utf-8 -a:basic
```

**OUTPUT:**

The method will return 0 for success or an integer for error or job in execution. An error such as the following can occur if an ISO image is not attached.

```
DetachISOImage_OUTPUT
   Message = ISO image is not attached
   MessageID = OSD32
   ReturnValue = 2
```

### 11.3.8 Boot To PXE

The `BootToPXE()` method is used to boot to server using the PXE mechanism, which is to reboot the host server and boot to PXE.

Invoke to boot target system to PXE with the following syntax:

**EXAMPLE:**

```plaintext
winrm i BootToPXE cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM_OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-encoding:utf-8 -a:basic
```

The return will be 0 for success or 1 if an error occurred in starting the processing the input parameters. The `MessageID` and `Message` output parameters will further contain method invocation information if an error occurred.

**OUTPUT:**

```
BootToPXE_OUTPUT
   ReturnValue = 0
```
11.3.9 Get Host MAC Address Information

Invoke GetHostMACInfo() with the following syntax:

EXAMPLE:

winrm i GetHostMACInfo cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM_OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman:443
-SkipCNCheck -encoding:utf-8 -a:basic

OUTPUT:

The return will be 0 for success and a list of MAC addresses or an integer for error or job in execution. The MessageID and Message output parameters will further contain method invocation information if an error occurred.

GetHostMACInfo_OUTPUT
MACList = 00219b927057, 00219b927059, 00219b92705b, 00219b92705d
ReturnValue = 0

11.3.10 Download ISO to VFlash

The DownloadISOToVFlash() method allows using remote command to download an ISO image to VFlash. The image needs to be an ISO image. Once this image is downloaded to VFlash, it can be booted via another WS-MAN command.

Invoke DownloadISOToVFlash() with the following parameters and syntax:

[IPADDRESS-ISO]: The IP address of the server that stores ISO images.

[DRIVESHARE]: This is the directory path to the ISO image.

[SHARETYPE]: The type of the remote storage. 0: NFS, 1: TFTP, 2: CIFS

[SHAREUSER]: User account for the ISO share location

[SHAREPASSWORD]: Password of the share account

[WORKGROUP]: Applicable workgroup

[IMAGENAME]: Image name of the iso image, such as boot.iso.

[Port]: Port number for connecting to the share, such as 2049.
EXAMPLE:

```
winrm i DownloadISOToVFlash cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+%SystemCreationClassName=DCIM_ComputerSystem
+%SystemName=DCIM:ComputerSystem
- u:[USER] - p:[PASSWORD]
- r:https://[IPADDRESS]/wsman:443 - encoding:utf-8 - a:basic @[IPAddress=[IPADDRESS-ISO];ShareName="/[DRIVEShare]";
ShareType=[SHARETYPE];Username=[SHAREUSER];
Password=[SHAREPASSWORD];Workgroup=[WORKGROUP];
ImageName=[IMAGENAME];Port=[PORT"]
```

OUTPUT:

The return will be 0 for success or 1 if an error occurred in starting the processing the input parameters. The MessageID and Message output parameters will further contain method invocation information if an error occurred.

```
DownloadISOToVFlash_OUTPUT
Job
  Address = http://schemas.xmlsoap.org/ws/2004/08
  /addressing/role/anonymous
ReferenceParameters
SelectorSet
  Selector: InstanceID = DCIM_OSDConcreteJob:1,
    __cimnamespace = root/dcim
  ReturnValue = 4096
```

The following error message is a direct result of a typo in the winRM input. Careful consideration must be applied to capitalization.

WSManFault

  Message = The WinRM client cannot process the request. The destination computer returned an empty response to the request.

Error number: -2144108299 0x803380F5

The WinRM client cannot process the request. The destination computer returned an empty response to the request.

11.3.11 Boot to ISO from VFlash

This method will expose the ISO Image present on VFlash as a CDROM device to the host server and boots to it.
Invoke **BootToISOFromVFlash()** with the following syntax:

**EXAMPLE:**

```
winrm i BootToISOFromVFlash
cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-SkipCNCheck -encoding:utf8 -a:basic
```

**OUTPUT:**

When this command is executed, a status or error message will be returned.

```
BootToISOFromVFlash_OUTPUT
Job
  Address = http://schemas.xmlsoap.org/ws/2004/08
    /addressing/role/anonymous
  ReferenceParameters
  SelectorSet
    Selector: InstanceID = DCIM_OSDConcreteJob:1,
      __cimnamespace = root/dcim
  ReturnValue = 4096
```

### 11.3.12 Delete ISO from VFlash

The **DeleteISOFromVFlash()** method will delete the ISO image that was downloaded to the **VFlash**.

Invoke **DeleteISOFromVFlash()** with the following syntax:

**EXAMPLE:**

```
winrm i DeleteISOFromVFlash
cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USERNAME] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-encoding:utf8 -a:basic
```

**OUTPUT:**

When this command is executed, a status or error message will be returned. If an image is not found, the following message will display:
11.3.13 Detach ISO from VFlash

The **DetachISOFromVFlash()** method will detach the ISO image in the *VFlash* from the system.

Invoke **DetachISOFromVFlash()** with the following syntax:

**EXAMPLE:**

```
winrm i DetachISOFromVFlash cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-encoding:utf-8 -a:basic
```

**OUTPUT:**

When this command is executed, a status or error message will be returned. If an image is not found the following message will display:

**DetachISOFromVFlash_OUTPUT**

```
Message = ISO Image not found on VFlash
MessageID = OSD41
ReturnValue = 2
```

11.3.14 Connect Network ISO Image

This method can be used to connect to a bootable ISO image located on a CIFS or NFS share. The ISO image is attached to the host system as an emulated USB CD-ROM storage device. Whenever the host system reboots it will boot to this ISO Image every single time until **DisconnectNetworkISOImage** is called. The ISO will be reattached upon iDRAC reset.

Invoke **ConnectNetworkISOImage()** via CIFS/NFS share with the following syntax:

- **[CIFS_or_NFS_IPADDRESS]**: This is the IP address of the location of the ISO image.
- **[/CIFS_or_NFS/OSISO]**: This is the sharename directory path to the ISO image.
- **[2_or_0]**: 2=CIFS, 0=NFS
[CIFS_or_NFS_Username]: This is the username to the IP address of the ISO image.

[CIFS_or_NFS_Password]: This is the corresponding password to the username containing the ISO image.

[OS.ISO]: This is to be replaced by the actual name of the ISO image.

**EXAMPLE:**

```
winrm i ConnectNetworkISOImage http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_OSDeploymentService ?CreationClassName=DCIM_OSDeploymentService +Name=DCIM:OSDeploymentService +SystemCreationClassName=DCIM:ComputerSystem +SystemName=DCIM:ComputerSystem -u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman:443 -SkipCNCheck -SkipCACheck -encoding:utf-8 -a:basic @[IPAddress="[CIFS_or_NFS_IPaddress]";ShareName="/[CIFS_or_NFS]";ShareType="[2_or_0]";Username="[CIFS_or_NFS_Username]"; Password="[CIFS_or_NFS_Password]";Workgroup="WORKGROUP"; ImageName="[OS.ISO]"
```

**OUTPUT:**

The return will be 0 for success or 1 if an error occurred in starting the processing the input parameters. The MessageID and Message output parameters will further contain method invocation information if an error occurred.

```
ConnectNetworkISOImage_OUTPUT
Job
  Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
SelectorSet
    Selector: InstanceID = DCIM_OSDConcreteJob:1,
      __cimnamespace = root/dcim
ReturnValue = 4096
```

**11.3.15 Disconnect Network ISO Image**

This method can be used to disconnect the target system from a bootable ISO image located on a CIFS or NFS share.

Invoke `DisconnectNetworkISOImage()` with the following syntax:
EXAMPLE:


OUTPUT:

The return will be 0 for success or 1 if an error occurred in starting the processing the input parameters. The MessageID and Message output parameters will further contain method invocation information if an error occurred.

DisconnectNetworkISOImage_OUTPUT
ReturnValue = 0

11.3.16 Skip ISO Image Boot

This method can be used to skip the target system from booting to a bootable ISO image (connected using ConnectNetworkISOImage method) one time only for next immediate host reboot. After that host server will continue to boot to the ISO image.

Invoke SkipISOImageBoot() via NFS share with the following syntax:

EXAMPLE:


OUTPUT:

Shown below are return messages of failure and success, 2 and 0, respectively. The MessageID and Message output parameters will further contain method invocation information if an error occurred.

Failure:

SkipISOImageBoot_OUTPUT
Message = ISO image is not attached
MessageID = OSD32
ReturnValue = 2
Success:

SkipISOImageBoot_OUTPUT
   ReturnValue = 0

11.3.17 Get Network ISO Image Connection Information

This method outputs the ISO connection status of the image that has been exposed to the host.

Invoke GetNetworkISOImageConnectionInfo() with the following syntax:

**EXAMPLE:**

```
winrm i GetNetworkISOImageConnectionInfo cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

GetNetworkISOImageConnectionInfo_OUTPUT
   Message = ISO image is not attached
   MessageID = OSD32
   ReturnValue = 2

11.3.18 Connect RFS ISO Image

The ConnectRFSISOImage() method is used to connect the ISO image that is mounted through Remote File Share (RFS) and is exposed to the host system as a USB-based CD-ROM device. The successful execution of this method shall connect to the ISO located on NFS/CIFS share to the host server and expose it as a virtual CDROM device using RFS USB endpoint. The successful execution of the method shall not change the boot order of that device. In order to boot to the CD-ROM, the CD-ROM shall be configured in the boot order in a separate step (using BIOS and Boot Management Profile), and the host server shall boot to the CD-ROM. Unlike the ConnectNetworkISOImage() method, the Lifecycle Controller is not locked and may perform other management tasks.

Invoke ConnectRFSISOImage() with the following syntax:

- **[IPADDRESS-ISO]:** The IP address of the server that stores ISO images.
- **[DRIVESHARE]:** This is the directory path to the ISO image.
- **[SHARETYPE]:** The type of the remote storage. 0: NFS, 2: CIFS
[SHAREUSER]: User account for the ISO share location

[SHAREPASSWORD]: Password of the share account

[WORKGROUP]: Applicable workgroup

[IMAGENAME]: Image name of the iso image, such as boot.iso.

**EXAMPLE:**

```powershell
winrm i ConnectRFSISOImage cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM_OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -a:basic @{IPAddress=[IPADDRESS];ShareName="/[DRIVESHARE]";ShareType="[SHARETYPE]";Username="[SHAREUSER]";Password="[SHAREPASSWORD]";Workgroup="[WORKGROUP]";ImageName="[IMAGENAME]"}
```

**OUTPUT:**

```powershell
ConnectRFSISOImage_OUTPUT
Job
   EndpointReference
       Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
   ReferenceParameters
       SelectorSet
           InstanceID = DCIM_OSDConcreteJob:1
           __cimnamespace = root/dcim
   ReturnValue = 4096
```

Concrete jobs return 4096 upon successful invocation. Poll for the concrete job “JobStatus = Success”.

**11.3.19 Disconnect RFS ISO Image**

The DisconnectRFSISOImage() method is used to disconnect and detach the ISO Image that is mounted through Remote File Share (RFS) and is exposed to the host system as a USB-based CD-ROM device.

Invoke DisconnectRFSISOImage() with the following syntax:

**EXAMPLE:**

```powershell
winrm i DisconnectRFSISOImage
```
11.3.20 Get RFS ISO Image Connection Information

The GetRFSISOImageConnectionInfo() method is used to provide the status of the ISO Image connection that has been exposed to the host system.

Invoke GetRFSISOImageConnectionInfo() with the following syntax:

EXAMPLE:

```
winrm i GetRFSISOImageConnectionInfo
cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic
```

OUTPUT:

GetRFSISOImageConnectionInfo_OUTPUT
Message = Unable to connect to ISO using RFS.
MessageID = OSD60
ReturnValue = 2

A return value of 0 indicates success, while the above output indicates an image was not present to retrieve the connection information from.

11.3.21 Boot To Hard Drive (HD)

The BootToHD() method is used for one time boot to the host server’s hard disk. After this method is executed the host is rebooted immediately and will boot to the first configured hard disk irrespective of its boot order.

Invoke BootToHD() with the following syntax:
EXAMPLE:

winrm i BootToHD cimv2/root/dcim/DCIM_OSDeploymentService
?CreationClassName=DCIM_OSDeploymentService
+Name=DCIM:OSDeploymentService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman:443
-encoding:utf-8 -a:basic

OUTPUT:

BootToHD_OUTPUT
  ReturnValue = 0

11.3.22 Configurable Boot to Network ISO

This method was added during the LC2 Version 1.1 release.

The ConfigurableBootToNetworkISO() works similar to BootToNetworkISO() except that the immediate boot to the ISO is not automatic and controlled by an input parameter called ResetType which will enable you to do a warm reset or cold reset or no immediate reset.

Invoke ConfigurableBootToNetworkISO() via NFS share with the following syntax:

[NFS_IPADDRESS]: This is the IP address of the location of the ISO image.

/[NFS/OSISO]: This is the directory path to the ISO image.

[NFS_Username]: This is the username to the IP address of the ISO image.

[NFS_Password]: This is the corresponding password to the username containing the ISO image.

[OS.ISO]: This is to be replaced by the actual name of the ISO image.

[RESET_TYPE]: 0=No reset, 1=warm reset 2=cold reset

EXAMPLE:

winrm i ConfigurableBootToNetworkISO
  cimv2/root/dcim/DCIM_OSDeploymentService
  ?CreationClassName=DCIM_OSDeploymentService
  +Name=DCIM:OSDeploymentService
  +SystemCreationClassName=DCIM_ComputerSystem
  +SystemName=DCIM:ComputerSystem
  -u: [USER] -p: [PASSWORD]
  -r:https://[IPADDRESS]/wsman:443 -SkipCNCheck
  -encoding:utf-8 -a:basic
  @[IPAddress="[NFS_IPaddress]";ShareName="[/NFS/OSISO]";ShareType="0";Username="[NFS_Username]";Password="[NFS_Password]";ResetType="[RESET_TYPE]";
OUTPUT:

The return will be 0 for success or 1 if an error occurred in starting the processing the input parameters. The MessageID and Message output parameters will further contain method invocation information if an error occurred.

ConfigurableBootToNetworkISO_OUTPUT

Job

Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
SelectorSet
Selector: InstanceID = DCIM_OSDConcreteJob:1,
__cimnamespace = root/dcim
ReturnValue = 4096

12 Lifecycle Controller Management Profile

The LC Management Profile describes the LC attribute configuration service and the collections and attributes instances that the service manages. The profile also describes the relationship of the LC attribute service to the DMTF/Dell profile version information and Dell Job Control profile.

The Dell Common Information Model (CIM) class extensions for supporting Lifecycle Controller feature management are defined in the Dell LC Management and related MOF files. The diagrams representing the classes that are implemented by the Lifecycle Controller firmware can be found in Dell LC Management Profile.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

12.1 Collect System Inventory on Restart (CSIOR)

By default, ‘collect system inventory on restart’ is disabled. To enable this feature, utilize the SetAttribute() method in the following example.

NOTE: To query the system to determine when the last CSIOR event occurred, list system inventory and examine the LastSystemInventoryTime attribute.

The Collect System Inventory on Restart attribute flags whether the system should do an automatic inventory or not. To get the current status of this attribute, see Section 12.3. The values can be:
- **Disabled** (default) = Disallow collecting inventory on restart
- **Enabled** = Allow collecting system inventory on restart

The **Part Firmware Update** attribute flags whether the Part Replacement automatic firmware update performed. The values can be:

- **Disable** (default) = firmware update is not allowed
- **Allow version upgrade only** = Allow firmware update only on up-revision
- **Match firmware of replaced part** = Always update firmware

The example below configures the **Part Replacement** feature to allow upgrade only and for the automatic synchronization to be on.

Invoke `setAttribute()` with the following parameters and syntax:

**EXAMPLE 1:**

```
winrm i SetAttribute http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService
-file:[DIRECTORYPATH]\SetAttribute_LC.xml
-r:https://[IPADDRESS]:443/wsman
-u:[USER] -p:[PASSWORD]
-auth:basic -encoding:utf-8
-SkipCNCheck -SkipCACheck
```

The input file `SetAttribute_LC.xml` is shown below:

```
<p:AttributeName>Part Firmware Update</p:AttributeName>
<p:AttributeValue>Allow version upgrade only</p:AttributeValue>
</p:SetAttribute_INPUT>
```

This method is used to set the values of multiple attributes.

Invoke `setAttributes()` with the following parameters and syntax:

**EXAMPLE 2:**

```
winrm i SetAttributes http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService
-file:[DIRECTORYPATH]\SetAttributes_LC.xml
```

The input file `SetAttributes_LC.xml` is shown below:

```
<p:AttributeName>Part Firmware Update</p:AttributeName>
<p:AttributeValue>Allow version upgrade only</p:AttributeValue>
</p:SetAttributes_INPUT>
```
The input file `SetAttributes_LC.xml` is shown below:

```xml
  <p:AttributeName>Part Firmware Update</p:AttributeName>
  <p:AttributeValue>Allow version upgrade only</p:AttributeValue>
  <p:AttributeName>Collect System Inventory on Restart</p:AttributeName>
  <p:AttributeValue>Enabled</p:AttributeValue>
</p:SetAttributes_INPUT>
```

**OUTPUT:**

SetAttribute_OUTPUT

```
RebootRequired = No
ReturnValue = 0
SetResult = Set PendingValue
```

### 12.2 Part Replacement Configuration and Management

If the `SetAttribute[s]()` method has been invoked, the pending values must be applied by creating a configuration job. The `CreateConfigJob()` method in the `DCIM_LCService` class creates a configuration job and executes it at the specified time.

#### 12.2.1 Create Config Job

Invoke `CreateConfigJob()` with the following parameters and syntax:

**EXAMPLE:**

```
winrm i CreateConfigJob http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem+Name=DCIM:LCService
-file:[DIRECTORYPATH]\CreateConfigJob.xml
-r:https://[IPADDRESS]:443/wsman
-u:[USER] -p:[PASSWORD] -auth:basic -encoding:utf-8
-SkipCNCheck -SkipCACheck
```

The input file `CreateConfigJob.xml` is shown below:

```xml
  <p:ScheduledStartTime>00000000002200.000000:000</p:ScheduledStartTime>
  <p:RebootIfRequired>false</p:RebootIfRequired>
</p:CreateConfigJob_INPUT>
```
The above command will schedule the job at 10pm. To poll for job completion, enumerate the `DCIM_LifecycleJob` job instance.

**OUTPUT:**

CreateConfigJob_OUTPUT

```
Job
  Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
SelectorSet
    Selector: InstanceID = JID_001265982202,
      __cimnamespace = root/dcim
Return Value = 0
```

To get the status of the above `jobID` or list all `jobIDs`, see **12.2.2** and **12.2.3**, respectively.

**12.2.2  Get LC Config Job Status**

**EXAMPLE:**

```
winrm g http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_LifecycleJob
?__cimnamespace=root/dcim
+InstanceID=JID_001265982202
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]:wsman -encoding:utf-8
-a:basic -SkipCNCheck -SkipCACheck
```

The method either returns a list of Concrete job objects or an error message. Check for the `JobStatus` property equal to `Completed` (shown below) to know the set has been completed.

**OUTPUT:**

```
DCIM_LifecycleJob
  InstanceID = JID_001265982202
  JobStartTime = 20191010101010
  JobStatus = COMPLETED
  JobUntilTime = 2009:8:11
  Message = The command was successful
  MessageArguments = null
  MessageID = LC001
  Name = LC Config
```

**12.2.3  List All LC Jobs**

**EXAMPLE:**
12.2.4 Get CSIOR Component Configuration Recovery (CCR) Attribute

The Component Configuration Recovery (CCR) attributes are:

- Licensed
- Part Firmware Update
- Collect System Inventory on Restart (CSIOR)
- Part Configuration Update

Get the current CSIOR attribute setting as follows:

**EXAMPLE 1:**

```bash
winrm g cimv2/root/dcim/DCIM_LCEnumeration
?InstanceID=LifecycleController.Embedded.1#LCAttributes.1#CollectSystemInventoryOnRestart
-u:[USERNAME] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman
-encoding:utf-8 -a:basic
```

NOTE: For 11G, InstanceID=DCIM_LCEnumeration:CCR5
OUTPUT:

DCIM_LCEnumeration
AttributeName = Collect System Inventory on Restart
CurrentValue = Disabled
DefaultValue = Enabled
ElementName = LC.emb.1
InstanceID = LifecycleController.Embedded.1#LCAttributes.1#CollectSystemInventoryOnRestart
IsReadOnly = false
PendingValue = null
PossibleValues = Enabled, Disabled

12.2.5 Get Part Firmware Update Attribute

Get the current Part Replacement firmware update mode as follows:

EXAMPLE:

winrm g http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCEnumeration
?InstanceID=LifecycleController.Embedded.1#LCAttributes.1#PartFirmwareUpdate
-u:[USERNAME] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman
-encoding:utf-8 -a:basic

NOTE: For 11G, InstanceID=DCIM_LCEnumeration:CCR4

OUTPUT:

DCIM_LCEnumeration
AttributeName = Part Firmware Update
CurrentValue = Allow version upgrade only
DefaultValue = Disable
ElementName = LC.emb.1
InstanceID = LifecycleController.Embedded.1#LCAttributes.1#PartFirmwareUpdate
IsReadOnly = false
PendingValue = null
PossibleValues = Disable, Allow version upgrade only, Match firmware of replaced part

See Section 12.5 to get the status on whether there is a valid VFlash License on the system.

12.3 Re-Initiate Auto-Discovery Client

Invoke the ReInitiateDHS() method to re-initialize and restart the Auto-Discovery client. All configuration information is replaced with the auto discovery factory defaults. Auto discovery can be disabled, enabled and initiated immediately, or delayed until next power cycle.

Invoke ReInitiateDHS() with the following parameters and syntax:
### [PS_IP_ADDRESS]
Substitution will need to be replaced with the actual IP address(s) or DNS name(s) of the Provisioning Server(s).

**PerformAutoDiscovery:**

1 = off (disables auto discovery)

2 = Now (enables and initiates auto discovery immediately)

3 = NextBoot (delay reconfiguration & auto discovery until next power cycle)

**EXAMPLE:**

```
winrm i ReInitiateDHS cimv2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem +CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem+Name=DCIM:LCService
-u:[USERNAME] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman
-encoding:utf-8 -a:basic -file:ReInitiateDHS.xml
```

The input file `ReInitiateDHS.xml` containing the parameters for the `ReInitiateDHS` method is shown below:

```
<p:ReInitiateDHS_INPUT xmlns:p="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService">
  <p:ResetToFactoryDefaults>TRUE</p:ResetToFactoryDefaults>
  <p:PerformAutoDiscovery>3</p:PerformAutoDiscovery>
</p:ReInitiateDHS_INPUT>
```

**OUTPUT:**

The output is status 0 for successfully set or an error message.

```
ReInitiateDHS_OUTPUT
  ReturnValue = 0
```

### 12.4 Clear or Set Provisioning Server

The Provisioning Server name (or a group names) can be cleared by invoking the `ClearProvisioningServer()` method on the `DCIM_LCService` class.

**Configuring the Provisioning Server name(s)**

**EXAMPLE-A:**

```
winrm i ClearProvisioningServer
cimv2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem +CreationClassName=DCIM_LCService
```
ClearProvisioningServer_OUTPUT
ReturnValue = 0

Setting the Provisioning Server name or IP address for the provisioning service

The Provisioning Server name and/or IP Addresses can be set by invoking the SetAttribute() method of the DCIM_LCService class.

[PS_IP_ADDRESS]: Substitution will need to be replaced with the actual IP address(s) or DNS name(s) of the Provisioning Server(s).

EXAMPLE-B:

winrm i SetAttribute
cimv2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem +CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem+Name=DCIM:LCService
-u:[USERNAME] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman
-encoding:utf-8 -a:basic

The input file SetProvisioningServer.xml is shown below:

```xml
  <p:AttributeName>Provisioning Server</p:AttributeName>
</p:SetAttribute_INPUT>
```

OUTPUT-B:

This method will return status 0 or error message.

SetAttribute_OUTPUT
RebootRequired = No
ReturnValue = 0
SetResult = Set CurrentValue
12.5 Check VFlash License Enablement

The following command can be used to check VFlash License enablement. Features such as Part Replacement, downloading ISO image to VFlash, or booting from VFlash are licensed features and require Dell VFlash SD Card to be inserted in order to function.

**EXAMPLE:**

```bash
winrm g cimv2/root/dcim/DCIM_LCEnumeration
?InstanceID=LifecycleController.Embedded.1#LCAttributes.1#Licensed
-u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman:443
-encoding:utf-8 -a:basic
```

**NOTE:** For 11G, InstanceID=DCIM_LCEnumeration:CCR1

**OUTPUT:**

This ‘get’ command will return the instance of the `DCIM_LCEnumeration` attribute class. The `CurrentValue` property will contain “True” (yes) or “False” (no) indicating whether features dependent on the presence of the VFlash SD card are enabled.

```
DCIM_LCEnumeration
  AttributeName = Licensed
  CurrentValue = Yes
  DefaultValue = No
  ElementName = LC.emb.1
  InstanceID = LifecycleController.Embedded.1#LCAttributes.1#Licensed
  IsReadOnly = true
  PendingValue
  PossibleValues = Yes, No
```

12.6 Download Server Public Key

This method is used to download the server public key to the Lifecycle Controller. A base64 encoded string containing the certificate authentication (CA) content is required as the input.

Invoke `DownloadServerPublicKey()` with the following parameters and syntax:

**EXAMPLE:**

```bash
winrm i DownloadServerPublicKey http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
?CreationClassName=DCIM_LCService
+Name=DCIM_LCService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-SkipCNCheck -SkipCACheck -encoding:utf-8
-a:basic -file:DownloadServerPublicKey.xml
```
The input file **DownloadServerPublicKey.xml** is shown below:

```
  <p:KeyContent>
    -----BEGIN CERTIFICATE-----
    MIIEQjCCA6ugAwIBAgIBADANBgkqhkiG9w0BAQQQFADCBzTELMAkGA1UEBhMCVVx
    CzAJBgNVBAgTAlRYMRQwEgYDVQQHEwtNYWluIFN0cmVldDEVMBMGA1UEChMMSm9l.
    .
    qvoMCKtoqLnGBByj/H2vyN7Fe/zMKXD5pO6XwYddGfA66w3HGUaR0+flKD40NDi9
    bKFEbRnZysUuKZ9c+RALZUIrvezemfX3fn1Yp7k05KU9vhY=
    -----END CERTIFICATE-----
  </p:KeyContent>
</p:DownloadServerPublicKey_INPUT>
```

**OUTPUT:**

When this method is executed, a **jobid** or an error message is returned. This **jobid** can then be used for subsequent processing with job control provider in **Section 10**.

DownloadServerPublicKey_OUTPUT

```
Job
  Address = http://schemas.xmlsoap.org/ws/2004/08
            /addressing/role/anonymous
  ReferenceParameters
  SelectorSet
    Selector: InstanceID = JID_001269440883, __cimnamespace = root/dcim
   ReturnValue = 0
```

### 12.7 Download Client Certificates

This method is used to download the client private certificate, password, and root certificate to Lifecycle Controller. A base64 encoded string containing the certificate authentication (CA) private key content is required as input.

Invoke **DownloadClientCerts()** with the following parameters and syntax:

**EXAMPLE:**

```
winrm i DownloadClientCerts http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
?CreationClassName=DCIM_LCService
+Name=DCIM:LCService
+SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -SkipCNCheck -SkipCACheck
```
The input file DownloadClientCerts.xml is shown below:

```xml
  <p:KeyContent>-----BEGIN RSA PRIVATE KEY-----
  Proc-Type: 4,ENCRYPTED
  DEK-Info: DES-EDE3-CBC,5FD6D6131DFA5A86
  uIG9hRg0lkoJkJmBkB5I8h5KnZkNUFnqPbQlNco9WzKyINR1FcIlIAU9ToUJOM
  Sn5S1A8fRBTfXzzVBA+KAt+34lvO/FEAijS0zKMW1nA+CUuzCFM7t3P+3kmD+o6a
  -----END RSA PRIVATE KEY-----
</p:KeyContent>
  <p:Password>[PASSWORD HERE]</p:Password>
  <p:CAContent>-----BEGIN CERTIFICATE-----
  MIIE2zCCA8OgAwIBAgIBADANBgkqhkiG9w0BAQQFADCBqTELMAkGA1UEBhMCVVMx
  CzAJBgNVBAgTAlRYMRQwEgYDVQQHEwtNYWluIFN0cmVldDEVMBMGA1UEChMMSmo5kZK8
  xcaSQ9UQKdH5z6sUasj8DYk6pXndgWIV5Wc9JfsN3+dratX3lrepoJPPhk
  N1htDxHYiDLwsg79yklJp1qZ5gdaeJ1jUYJB6hRDQ+X7HxWN2VNk+ZIvvyYzC=
  -----END CERTIFICATE-----
</p:CAContent>
</p:DownloadClientCerts_INPUT>

**OUTPUT:**

When this method is executed, a **jobid** or an error message is returned. This **jobid** can then be used for subsequent processing with job control provider in Section 10.

**12.8 Delete Auto-Discovery Client Certificates**

This method is used to delete the client certificates set previously by the auto discovery method.

Invoke **DeleteAutoDiscoveryClientCerts()** with the following parameters and syntax:
EXAMPLE:

winrm i DeleteAutoDiscoveryClientCerts

http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService
-u:%USERNAME% -p:%PASSWORD%
-r:https://%IPADDRESS%/wsman
-encoding:utf-8 -a:basic -SkipCASet -SkipCNCheck -skiprevocationcheck

OUTPUT:
DeleteAutoDiscoveryClientCerts_OUTPUT
Returnvalue = 0

12.9 Set Public Certificates
This method is used to update a public SSL Certificate on the iDRAC.

Invoke SetPublicCertificate() with the following parameters and syntax:

Type: Specifies certificate service
directoryCA = certificate for Active Directory or LDAP server

EXAMPLE:

?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic 
-file:SetPublicCertificate.xml

The input file SetPublicCertificate.xml is shown below:

<p:Type>directoryCA</p:Type>
<p:Certificate>
-----BEGIN CERTIFICATE-----
MIID9DCCA12gAwIBAgIBADANBgkqhkiG9w0BAQQFADCBszELMAkGA1UEBhMCVVMx
CzAJBgNVBAgTAlRYMQ8wDQYDVQQHEwZBdXN0aW4xFjAU.
.M.
H/ea71Ltb/Au2QFhqcHkeUEbQ4qXSXTmDEgKAImKjoCAaWHcDqEwvUcxGl4ekG
LaUEGQhQicLe+03RDp05j+yP0iv/N10OGMfthWg/LJ3EoV1Zba2tXnCp8XxCukJC
12.10 Set iDRAC Certificate and Private Key

This method is used to update an iDRAC certificate and private key pairs using the contents of a PKCS#12 file.

Invoke SetCertificateAndPrivateKey() with the following parameters and syntax:

- **Type**: Specifies the service the certificate is for:
  - server = web server

- **PKCS12**: Represents the base64 encoded contents of PKCS#12 file to upload. Note this is the contents of the file and not a filename.

- **PKCS12pin**: Password to decode the PKCS12

**EXAMPLE:**

```
winrm i SetCertificateAndPrivateKey http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file: SetCertificateAndPrivateKey.xml
```

The input file SetCertificateAndPrivateKey.xml is shown below:

```
  <p:Type>server</p:Type>
  <p:PKCS12>MIIPUQIBAzCCDxcGCSqGSIb3DQEHAaCCDwgEgg8EMIIPADCCBTcGCSqGSIb3DQEH
BqCCBSgwgUkAgEAMIIFHQYJKoZIhvcNAQcBMBwGCiqGSIb3DQEMARYwDgQIySf0
. .
. .
CSqGSIb3DQEJFTEWBBQYcErnuOyBo9ayA3cswSZO6x70NTAxMCEwCQYFKw4DAhoF
AAQU+yOoD76JK1t4yzDgnOE562Cv9AQECM9hIYFEGiLAgIYAA==
</p:PKCS12>
</p:SetCertificateAndPrivateKey_INPUT>
```
12.10 Set Certificate and Private Key

This method is used to set the certificate and private key for secure communication.

Invoke SetCertificateAndPrivateKey() with the following parameters and syntax:

```
winrm i SetCertificateAndPrivateKey

cimv2/root/dcim/DCIM_LCService

?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService

-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman
-encoding:utf-8 -a:basic -SkipCACheck -SkipCNCheck
```

OUTPUT:

SetCertificateAndPrivateKey_OUTPUT

Message = Server certificate successfully modified, iDRAC will now reset and be unavailable for a few minutes

MessageID = LC018

ReturnValue = 0

12.11 Delete Auto-Discovery Server Public Key

This method is used to delete the public server key set previously by the set auto discovery method.

Invoke DeleteAutoDiscoveryServerPublicKey() with the following parameters and syntax:

```
winrm i DeleteAutoDiscoveryServerPublicKey

cimv2/root/dcim/DCIM_LCService

?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService

-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman
-encoding:utf-8 -a:basic -SkipCACheck -SkipCNCheck
```

OUTPUT:

DeleteAutoDiscoveryServerPublicKey_OUTPUT

ReturnValue = 0

12.12 Insert Comment in Lifecycle Controller Log

This method is used to insert additional user comments into the Lifecycle Controller log.

Invoke InsertCommentInLCLog() with the following parameters and syntax:

```
winrm i InsertCommentInLCLog

http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService

?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService

-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCACheck
```

OUTPUT:

InsertCommentInLCLog_OUTPUT

ReturnValue = 0
The input file `InsertCommentInLCLog.xml` is shown below:

```xml
<p:InsertCommentInLCLog_INPUT xmlns:p="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService">
  <p:Comment>INSERT COMMENT HERE</p:Comment>
</p:InsertCommentInLCLog_INPUT>
```

**OUTPUT:**

```
InsertCommentInLCLog_OUTPUT
  ReturnValue = 0
```

## 12.13 Export Lifecycle Controller Log

This method is used to export the log from the Lifecycle Controller after processing jobs.

Invoke `ExportLCLog()` with the following parameters and syntax:

- **IPAddress**: This is the IP address of the target export server.
- **ShareName**: This is the directory path to the mount point.
- **FileName**: This is the target output file.
- **ShareType**: Type of share
  - NFS=0, CIFS=2
- **Username**: This is the username to the target export server.
- **Password**: This is the password to the target export server.
- **Workgroup**: This is the applicable workgroup.

**EXAMPLE:**

```
winrm i ExportLCLog http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:ExportLCLog.xml
```

The input file `ExportLCLog.xml` is shown below:

```xml
  <p:IPAddress>123.456.7.8</p:IPAddress>
</p:ExportLCLog_INPUT>
```
When this method is executed, a `jobid` or an error message is returned.

**EXAMPLE:**

```
winrm i ExportHWInventory
```

```
http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
```

```
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
```
The input file `ExportHWInventory.xml` is shown below:

```xml
  <p:IPAddress>123.456.7.8</p:IPAddress>
  <p:ShareName>sharename</p:ShareName>
  <p:FileName>filename.txt</p:FileName>
  <p:ShareType>0</p:ShareType>
  <p:Username>admin</p:Username>
  <p:Password>password</p:Password>
  <p:Workgroup>workgroup</p:Workgroup>
</p:ExportHWInventory_INPUT>
```

**OUTPUT:**

When this method is executed, a `jobid` or an error message is returned.

ExportHWInventory_OUTPUT

Job

  Address = http://schemas.xmlsoap.org/ws/2004/08/addressing
            /role/anonymous

ReferenceParameters


SelectorSet

  Selector: InstanceID = JID_001271167557, __cimnamespace = root/dcim

ReturnValue = 0

**12.15 Export Factory Configuration**

This method is used to export the factory configuration from the Lifecycle Controller to a text file on a remote share.

Invoke `ExportFactoryConfiguration()` with the following parameters and syntax:

- **IPAddres**: This is the IP address of the target export server.
- **ShareName**: This is the directory path to the mount point.
- **FileName**: This is the target output file.
- **ShareType**: Type of share
  - NFS=0, CIFS=2
- **Username**: This is the username to the target export server.
- **Password**: This is the password to the target export server.
Workgroup: This is the applicable workgroup.

EXAMPLE:

```bash
winrm i ExportFactoryConfiguration http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem+Name=DCIM:LCService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:ExportFactoryConfiguration.xml
```

The input file `ExportFactoryConfiguration.xml` is shown below:

```xml
  <p:IPAddress>123.456.7.8</p:IPAddress>
  <p:ShareName>sharename</p:ShareName>
  <p:FileName>filename.txt</p:FileName>
  <p:ShareType>0</p:ShareType>
  <p:Username>admin</p:Username>
  <p:Password>password</p:Password>
  <p:Workgroup>workgroup</p:Workgroup>
</p:ExportFactoryConfiguration_INPUT>
```

OUTPUT:

When this method is executed, a jobid or an error message is returned.

ExportFactoryConfiguration_OUTPUT

```
Job Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
  SelectorSet
    Selector: InstanceID = JID_001271168441, __cimnamespace = root/dcim
ReturnValue = 0
```

12.16 System Decommission

This method is called to delete all configurations from the Lifecycle controller before the system is retired.

Invoke `LCWipe()` with the following parameters and syntax:

EXAMPLE:

```bash
winrm i LCWipe http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
```

104
12.17 Get Remote Services API Status

The GetRemoteServicesAPIStatus() method is used to obtain the overall remote services API status that includes both the host system status as well as the remote services (Data Manager) status. The overall rolled up status shall be reflected in the Status output parameter.

**NOTE:** The LCStatus output parameter value includes the status reported by the DMStatus output parameter in the GetRSSstatus() method. Thus, GetRSSstatus() method invocation is redundant.

Invoke `GetRemoteServicesAPIStatus()` with the following parameters and syntax:

**EXAMPLE:**

```powershell
 SystemCreationClassName=DCIM_ComputerSystem
 CreationClassName=DCIM_LCService
 SystemName=DCIM:ComputerSystem
 Name=DCIM:LCService
-u:[USER] -p:[PASSWORD]
 -r:[IPADDRESS]/wsman -SkipCNcheck -SkipCACheck
 -encoding:utf-8 -a:basic
```

**OUTPUT:**

```
GetRemoteServicesAPIStatus_OUTPUT
 LCStatus = 0
 Message = Lifecycle Controller Remote Services is ready.
 MessageID = LC061
 ReturnValue = 0
 ServerStatus = 2
 Status = 0
```

12.18 Export System Configuration

This method is used to export the system configuration from the Lifecycle Controller to a file on a remote share.

Invoke `ExportSystemConfiguration()` with the following parameters and syntax:
**IPAddress**: This is the IP address of the target export server.

**ShareName**: This is the directory path to the mount point.

**FileName**: This is the target output file.

**ShareType**: Type of share

- NFS=0, CIFS=2

**Username**: This is the username to the target export server.

**Password**: This is the password to the target export server.

**EXAMPLE:**

```plaintext
winrm i ExportSystemConfiguration http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService?SystemCreationClassName=DCIM_ComputerSystem+CreationClassName=DCIM_LCService+SystemName=DCIM:ComputerSystem+Name=DCIM:LCService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic @{IPAddress="SHARE_IP_ADDRESS"; ShareName="SHARE_NAME"; ShareType="SHARE_TYPE"; FileName="SHARE_OUTPUT_FILE_NAME"; Username="SHARE_USERNAME"; Password="SHARE_PASSWORD"}
```

**OUTPUT:**

When this method is executed, a *jobid* or an error message is returned.

ExportSystemConfiguration_OUTPUT

- SelectorSet Selector: InstanceID = JID_001271168441, __cimnamespace = root/dcim
- ReturnValue = 0

**12.19 Import System Configuration**

This method is used to import the system configuration from the Lifecycle Controller from a file on a remote share.

Invoke *ImportSystemConfiguration()* with the following parameters and syntax:

- **IPAddress**: This is the IP address of the target export server.

- **ShareName**: This is the directory path to the mount point.
FileName: This is the target output file.

ShareType: Type of share
          NFS=0, CIFS=2

Username: This is the username to the target export server.

Password: This is the password to the target export server.

EXAMPLE:

winrm i ImportSystemConfiguration http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService?SystemCreationClassName=DCIM_ComputerSystem+CreationClass
Name=DCIM_LCService+SystemName=DCIM:ComputerSystem+Name=DCIM:LCService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACHcheck
-encoding:utf-8 -a:basic @{IPAddress="SHARE_IP_ADDRESS";ShareName="SHARE_NAME";ShareType="SHARE_TYPE";FileName="SHARE_OUTPUT_FILE_NAME";Username="SHARE_USERNAME";Password="SHARE_PASSWORD"}

OUTPUT:

When this method is executed, a jobid or an error message is returned.

ImportSystemConfiguration_OUTPUT
    Job
        Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    ReferenceParameters
    SelectorSet
        Selector: InstanceID = JID_001271168441, __cimnamespace = root/dcim
    ReturnValue = 0

13 VFlash SD Card Management

The Persistent Storage Profile describes the necessary properties and methods for representing and managing the partitions on the virtual flash media (SD Card on AMEA) provided by the iDRAC in Dell platforms.

The partition management of the virtual flash media includes:
- Listing virtual flash partitions
- Creating new partitions
- Deleting existing partitions
- Formatting a partition
- Exposing the partition in the host OS
- Detaching an attached partition
- Uploading an image to a partition
- Booting to a partition
- Modifying a partition
- Copying/exporting the contents of the partition

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

13.1 Listing the SD Card Partitions
Each partition on the virtual flash media shall be represented by an instance of
DCIM_OpaqueManagementData. If nothing is returned, no partitions exist. Use the CreatePartition() method to create partitions.

Enumerate the DCIM_OpaqueManagementData with the following parameters and syntax:

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_OpaqueManagementData
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -SkipCNcheck -SkipCACheck -encoding:utf-8 -a:basic

OUTPUT:

DCIM_OpaqueManagementData
   AccessType = Read Only
   AttachedState = Detach
   CreationClassName = DCIM_OpaqueManagementData
   DataFormat = RAW
   DeviceID = DCIM_OpaqueManagementData:Partition1
   ElementName = VFlash
   Name = label1
   PartitionIndex = 1
   PartitionType = HDD
   Size = 50
   SystemCreationClassName = DCIM_ComputerSystem
   SystemName = DCIM:ComputerSystem

Note: If nothing is returned, no partitions exist. Use the CreatePartition method to create partitions.

13.2 Initialize the Virtual Flash Media

- Enumerate the DCIM_PersistentStorageService class
- Invoke the InitializeMedia method on the instance above
- The OUT parameter Job will refer to the instance of CIM_ConcreteJob using which the user can query the status of the initialization of the media.
13.2.1 Get VFlash SD Card Inventory

DCIM_VFlashView is a subclass of CIM_View that is used to represent the physical attributes of the virtual flash media, such as total size, available size, category etc. on which the partitions will reside.

Enumerate the DCIM_VFlashView with the following parameters and syntax:

**EXAMPLE:**

```bash
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_VFlashView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

DCIM_VFlashView

- AvailableSize = 972
- Capacity = 972
- ComponentName = vFlash SD Card
- FQDD = Disk.vFlashCard.1
- HealthStatus = OK
- InitializedState = Uninitialized
- InstanceID = Disk.vFlashCard.1
- LastSystemInventoryTime = 20100426221347.000000+000
- LastUpdateTime = 20100426221347.000000+000
- Licensed = true
- VFlashEnabledState = true
- WriteProtected = false

**InitializedState:** Field indicates status of element to be initialized

**InstanceId:** InstanceID of desired element for initialization

13.2.2 Initialize / Format Media

This method is used to initialize or format the virtual flash media device.

Invoke InitializeMedia() with the following parameters and syntax:

**EXAMPLE:**

```bash
winrm i InitializeMedia http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_PersistentStorageService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_PersistentStorageService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:PersistentStorageService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

See Section 13.2.3 for the populated initialized fields
When this method is executed, a `jobid` or an error message is returned.

 InitializeMedia_OUTPUT
  Job
  Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
  ReferenceParameters
    ResourceURI = http://schemas.dell.com/wbem
    SelectorSet
      Selector: InstanceID = JID_001268732835,
      __cimnamespace = root/dcim
  ReturnValue = 0

13.2.3 Verify Initialization / Formatting
After invoking InitializeMedia(), get the instance of DCIM_VFlashView to confirm successful initialization.

Get a specific DCIM_VFlashView with the following parameters and syntax:

```
[INSTANCE_ID] = Obtained from Section 13.2.1, such as Disk.vFlashCard.1
```

**EXAMPLE:**

```shell
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

DCIM_VFlashView
  AvailableSize = 972
  Capacity = 972
  ComponentName = vFlash SD Card
  FQDD = Disk.vFlashCard.1
  HealthStatus = OK
  InitializedState = Initialized
  InstanceID = Disk.vFlashCard.1
  LastSystemInventoryTime = 20100426221347.000000+000
  LastUpdateTime = 20100426221347.000000+000
  Licensed = true
  VFlashEnabledState = true
  WriteProtected = false

**InitializedState:** Field indicates status of element to be initialized

**InstanceId:** InstanceID of desired element for initialization

See Section 13.2.1 for the populated uninitialized fields
13.3 Enable/Disable VFlash using VFlash State Change

This method is used to enable or disable the virtual flash media device. When the VFlashStateChange() method is successfully executed, the change will be dictated in the VFlashEnabledState parameter as shown in Section 13.2.1 and Section 13.2.3.

Invoke VFlashStateChange() with the following parameters and syntax:

**RequestedState**: The state to set to

Enable=1, Disable=2

**EXAMPLE**:

```
winrm i VFlashStateChange http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_PersistentStorageService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_PersistentStorageService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:PersistentStorageService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:VFlashStateChange.xml
```

The input file VFlashStateChange.xml is shown below:

```
  <p:RequestedState>1</p:RequestedState>
</p:VFlashStateChange_INPUT>
```

**OUTPUT**:

```
VFlashStateChange_OUTPUT
  ReturnValue = 0
```

13.4 Create Partition

This method is used for creating a new partition on a storage device. When this method is successfully executed, an instance of DCIM_OpaqueManagementData representing the desired partition will be created (Section 13.1) and a reference to this instance is captured in the output parameter Job.

Invoke CreatePartition() with the following parameters and syntax:

**PartitionIndex**: The PartitionIndex property of the DCIM_OpaqueManagementData instance that represents the partition to be formatted

1 to 16

**Size**: The size of the partition to be created

**SizeUnit**: The unit of the size
MB=1, GB=2

**PartitionType**: The partition type

floppy=1, hard disk=2

**OSVolumeLabel**: The label seen in the OS after attaching the partition

**EXAMPLE**:

```
winrm i CreatePartition http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_PersistentStorageService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_PersistentStorageService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:PersistentStorageService
-r:https://[IPADDRESS]:443/wsm
-u:[USER] -p:[PASSWORD] -a:basic
-e:utf-8 -s:SkipCNCheck -s:SkipCACheck
-f:[DIRECTORYPATH]\CreatePartition.xml
```

The input file `CreatePartition.xml` is shown below:

```xml
    <p:PartitionIndex>1</p:PartitionIndex>
    <p:Size>50</p:Size>
    <p:SizeUnit>1</p:SizeUnit>
    <p:PartitionType>2</p:PartitionType>
    <p:OSVolumeLabel>label1</p:OSVolumeLabel>
</p:CreatePartition_INPUT>
```

**OUTPUT**:

When this method is executed, a *jobid* or an error message is returned.

CreatePartition_OUTPUT

```
Job
Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
SelectorSet
    Selector: InstanceID = JID_001270734913, __cimnamespace = root/dcim
ReturnValue = 0
```

If this method returns the following message, the *VFlash* must be enabled using the `VFlashStateChange()` ([Section 13.3](#)) method.

CreatePartition_OUTPUT

```
Message = VFlash not enabled
MessageID = VF015
ReturnValue = 2
```
13.5 Create Partition using Image

This method creates a partition on the storage device using the image provided by the user. The partition size will be the same as the size of the image. The maximum size of the image is 4GB.

The image can be located on a NFS/CIFS share or on a TFTP server. When this method is successfully executed, an instance of DCIM_OpaqueManagementData representing the desired partition will be created (Section 13.1), and a reference to this instance is captured in the output parameter Job.

Invoke CreatePartitionUsingImage() with the following parameters and syntax:

- **PartitionIndex**: The PartitionIndex property of the DCIM_OpaqueManagementData instance that represents the partition to be formatted
  - 1 to 16

- **PartitionType**: The format types that these partitions need to be formatted as
  - floppy=1, hard disk=2, CD ROM=3

- **OSVolumeLabel**: The label seen in the OS after attaching the partition

- **URI**: The URI location of firmware to update a component
  - Supported protocols are FTP and HTTP.

- **IPAddress**: IP address of TFTP or NFS share

- **ShareType**: Type of share
  - NFS=0, TFTP=1, CIFS=2, FTP=3, HTTP=4

- **SharePath**: NFS sharepoint address

- **ImageName**: Name of the ISO or IMG image

- **Workgroup**: Name of the workgroup, if applicable

- **Username**: The username to be used to access the file

- **Password**: The password to be used to access the file

- **Port**: The port number to be used

- **HashType**: The hash type
  - MD5=1, SHA1=2

- **HashValue**: The hash value string based on the HashType parameter

**EXAMPLE:**
winrm i CreatePartitionUsingImage http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_PersistentStorageService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_PersistentStorageService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:PersistentStorageService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:CreatePartitionUsingImage.xml

The input file CreatePartitionUsingImage.xml is shown below:

```xml
  <p:PartitionIndex>1</p:PartitionIndex>
  <p:PartitionType>2</p:PartitionType>
  <p:OSVolumeLabel>label</p:OSVolumeLabel>
  <p:URI>ftp://123.456.7.89/dir/filename.exe</p:URI>
  <p:IPAddress>123.456.7.8</p:IPAddress>
  <p:ShareType>3</p:ShareType>
  <p:SharePath></p:SharePath>
  <p:ImageName>imagename.iso</p:ImageName>
  <p:Workgroup>workgroup</p:Workgroup>
  <p:Username>Administrator</p:Username>
  <p:Password>password</p:Password>
  <p:Port></p:Port>
  <p:HashType>1</p:HashType>
  <p:HashValue>123</p:HashValue>
</p:CreatePartitionUsingImage_INPUT>

OUTPUT:

When this method is executed, a \textit{jobid} or an error message is returned.

CreatePartitionUsingImage\_OUTPUT
\begin{itemize}
  \item \textbf{Job}
  \begin{itemize}
    \item Address = \url{http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous}
    \item ReferenceParameters
    \begin{itemize}
      \item ResourceURI = \url{http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_LifecycleJob}
      \item SelectorSet
      \begin{itemize}
        \item Selector: InstanceID = JID\_001268833219, __cimnamespace = root/dcim
      \end{itemize}
    \end{itemize}
    \item ReturnValue = 0
  \end{itemize}
\end{itemize}

Reference Section 13.2 to fix an uninitialized media device error:

CreatePartitionUsingImage\_OUTPUT
\begin{itemize}
  \item \textbf{Message} = SD card not initialized
  \item MessageID = VF017
  \item ReturnValue = 2
\end{itemize}
13.6 Delete Partition

This method is for deleting a partition on a storage device. When this method is successfully executed, the instance of `DCIM_OpaqueManagementData` representing the desired partition along with the association instance of `DCIM_ServiceAffectsElement` will be deleted. The `AvailableSize` property of the associated storage media will increase by the size of the deleted partition.

Invoke `DeletePartition()` with the following parameters and syntax:

```
PartitionIndex: The PartitionIndex property of the DCIM_OpaqueManagementData instance that represents the partition to be removed
1 to 16
```

**EXAMPLE:**

```
winrm i DeletePartition http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_PersistentStorageService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_PersistentStorageService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:PersistentStorageService
-r:https://[IPADDRESS]:443/wsman
-u:[USER] -p:[PASSWORD] -auth:basic
-encoding:utf-8 -SkipCNCheck -SkipCACheck
-file:[DIRECTORYPATH]\DeletePartition.xml
```

The input file `DeletePartition.xml` is shown below:

```
<p:PartitionIndex>1</p:PartitionIndex>
</p:DeletePartition_INPUT>
```

**OUTPUT:**

When this method is executed, a `ReturnValue` or error message is returned.

```
DeletePartition_OUTPUT
  ReturnValue = 0
```

An index that does not exist in the XML file may yield the following error message:

```
DeletePartition_OUTPUT
  Message = Invalid partition index
  MessageID = VF018
  ReturnValue = 2
```

13.7 Format Partition

This method is for formatting a partition of the type specified by the user.
Use the following algorithm to successfully format an existing partition:

- **Enumerate the** `DCIM_PersistentStorageService` **class**

- **Invoke the** `FormatPartition()` **method on the instance above with the following parameters:**
  
  **PartitionIndex:** The `PartitionIndex` property of the `DCIM_OpaqueManagementData` instance that represents the partition to be formatted
  
  1 to 16

  **FormatType:** The new format type of the partition
  
  EXT2=1, EXT3=2, FAT16=3, FAT32=4

- The OUT parameter Job will refer to the instance of `CIM_ConcreteJob` using which the user can query the status of the formatting of the partition.

**EXAMPLE:**

```bash
winrm i FormatPartition http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_PersistentStorageService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_PersistentStorageService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:PersistentStorageService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:FormatPartition.xml
```

The input file `FormatPartition.xml` is shown below:

```xml
<p:PartitionIndex>13</p:PartitionIndex>
<p:FormatType>4</p:FormatType>
</p:FormatPartition_INPUT>
```

**OUTPUT:**

When this method is executed, a `jobid` or an error message is returned.

```xml
FormatPartition_OUTPUT

Job
  Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
  ReferenceParameters
  SelectorSet
    Selector: InstanceID = JID_001270738393, __cimnamespace = root/dcim
  ReturnValue = 0
```
13.8 Modify Partition

This method is used for modifying the changeable attributes of a partition.

Use the following algorithm to successfully modify an existing partition.

- Enumerate the `DCIM_PersistentStorageService` class
- Invoke `ModifyPartition()` method on the instance above with the following parameters:

  - **PartitionIndex**: The `PartitionIndex` property of the `DCIM_OpaqueManagementData` instance that represents the partition to be modified
    - 1 to 16
  - **AccessType**: The type of access level
    - Read-Only=1, Read-Write=3
- The OUT parameter Job will refer to the instance of `CIM_ConcreteJob` using which the user can query the status of the modification of the partition.

**EXAMPLE**:

```
winrm i ModifyPartition http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_PersistentStorageService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_PersistentStorageService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:PersistentStorageService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAtch
-encoding:utf-8 -a:basic -file:ModifyPartition.xml
```

The input file `ModifyPartition.xml` is shown below:

```
  <p:PartitionIndex>6</p:PartitionIndex>
  <p:AccessType>3</p:AccessType>
</p:ModifyPartition_INPUT>
```

**OUTPUT**:

```
ModifyPartition_OUTPUT
  ReturnValue = 0
```

13.9 Attach Partition

This method is for defining the set of partitions to be exposed as Floppy/CD/HDD endpoints to the managed system and BIOS.
Invoke AttachPartition() with the following parameters and syntax:

**PartitionIndex:** The PartitionIndex property of the DCIM_OpaqueManagementData instance that represents the partition to be attached

1 to 16

**EXAMPLE:**

winrm i AttachPartition http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_PersistentStorageService
\?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_PersistentStorageService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:PersistentStorageService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:AttachPartition.xml

The input file AttachPartition.xml is shown below:

```xml
  <p:PartitionIndex>12</p:PartitionIndex>
</p:AttachPartition_INPUT>
```

**OUTPUT:**

When this method is executed, a jobid or an error message is returned.

AttachPartition_OUTPUT

Job
  Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
  ReferenceParameters
  SelectorSet
    Selector: InstanceID = JID_001270737179, __cimnamespace = root/dcim
  ReturnValue = 0

13.10 Detach Partition

This method is for defining the set of partitions to be removed as USB endpoints from the managed system.

Invoke DetachPartition() with the following parameters and syntax:

**PartitionIndex:** The PartitionIndex property of the DCIM_OpaqueManagementData instance that represents the partition to be detached

1 to 16

**EXAMPLE:**
winrm i DetachPartition http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_PersistentStorageService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_PersistentStorageService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:PersistentStorageService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman
-SkipCNcheck -SkipCAcheck
-encoding:utf-8
-a:basic -file:DetachPartition.xml

The input file DetachPartition.xml is shown below:

    <p:PartitionIndex>12</p:PartitionIndex>
</p:DetachPartition_INPUT>

OUTPUT:

When this method is executed, a jobid or an error message is returned.

DetachPartition_OUTPUT
    Job
        Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
        ReferenceParameters
            SelectorSet
                Selector: InstanceID = JID_001270737364, __cimnamespace = root/dcim
        ReturnValue = 0

If the partition is already detached, the following message may be displayed:

DetachPartition_OUTPUT
    Message = Partition already detached
    MessageID = VF028
    ReturnValue = 2

13.11 Export Data from Partition

This method is for exporting the contents of a partition to a location specified by the user.

Use the following algorithm to successfully export data from an existing partition.

- Enumerate the DCIM_PersistentStorageService class
- Invoke the ExportDataFromPartition() method on the instance above with the following parameters:

  PartitionIndex: The PartitionIndex property of the DCIM_OpaqueManagementData instance that represents the partition to be formatted

    1 to 16

  IPAddress: IP address of TFTP or NFS share
**ShareType**: Type of share  
NFS=0, TFTP=1, CIFS=2

**SharePath**: NFS sharepoint address

**ImageName**: Name of the ISO or IMG image

**Workgroup**: Name of the workgroup, if applicable

**Username**: The username to be used to access the file

**Password**: The password to be used to access the file

**Port**: The port number to be used

**HashType**: The hash type  
MD5=1, SHA1=2

**HashValue**: The hash value string based on the *HashType* parameter

**EXAMPLE**:  
```
```

The input file *ExportDataFromPartition.xml* is shown below:

```xml
  <p:PartitionIndex>1</p:PartitionIndex>
  <p:IPAddress>123.456.7.8</p:IPAddress>
  <p:ShareType>2</p:ShareType>
  <p:SharePath>/temp</p:SharePath>
  <p:ImageName>imagename.iso</p:ImageName>
  <p:Workgroup>workgroup</p:Workgroup>
  <p:Username>Administrator</p:Username>
  <p:Password>password</p:Password>
  <p:Port/></p:Port>
  <p:HashType>1</p:HashType>
  <p:HashValue>123</p:HashValue>
</p:ExportDataFromPartition_INPUT>
```

**OUTPUT**:  
```
```
When this method is executed, a jobid or an error message is returned.

ExportDataFromPartition_OUTPUT

Job

Address = http://schemas.xmlsoap.org/ws/2004/08/addressing
         /role/anonymous

ReferenceParameters


SelectorSet

Selector: InstanceID = JID_001271681930, __cimnamespace = root/dcim

ReturnValue = 0

14 Boot Control Configuration Management

This feature provides the ability to get and set the boot order configuration. The Boot Control Profile describes the classes, associations, properties, and methods used to manage the boot control configurations of a physical or virtual computer system.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

14.1 Listing the Boot Inventory-ConfigSetting Class

The boot configuration settings are a collection of settings that are applied to the boot configurable system during the boot process. The current, default, and next status fields of each element are available.

Enumerate BootConfigSetting with the following parameters and syntax:

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BootConfigSetting
-u:USER -p:PASSWORD
-r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic

OUTPUT:

DCIM_BootConfigSetting
   ElementName = BootSeq
   InstanceID = IPL
   IsCurrent = 1
   IsDefault = 0
   IsNext = 1

DCIM_BootConfigSetting
   ElementName = HddSeq

This InstanceID can be used as input for a 'get' operation, as shown in Section 14.2
InstanceID = BCV
IsCurrent = 2
IsDefault = 0
IsNext = 2

DCIM_BootConfigSetting
  ElementName = UefiBootSeq
  InstanceID = UEFI
  IsCurrent = 2
  IsDefault = 0
  IsNext = 2

DCIM_BootConfigSetting
  ElementName = OneTimeBootMode
  InstanceID = OneTime
  IsCurrent = 2
  IsDefault = 0
  IsNext = 2

DCIM_BootConfigSetting
  ElementName = vFlash Boot Configuration
  InstanceID = vFlash
  IsCurrent = 2
  IsDefault = 0
  IsNext = 2

14.2 Getting a Boot ConfigSetting Instance

Getting the boot configuration current, default, and next attributes of one particular boot configuration instance is an alternative to enumerating all available instances as shown in Section 14.1.

Get a BootConfigSetting instance with the following parameters and syntax:

[INSTANCEID]: This is obtained from the enumeration in Section 14.1, in which this example would use IPL as an instance ID

EXAMPLE:

winrm g http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BootConfigSetting
?InstanceID=[INSTANCEID]
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic

OUTPUT:

DCIM_BootConfigSetting
  DCIM_BootConfigSetting
  BootConfigSetting
  ElementName = BootSeq
  InstanceID = IPL
  IsCurrent = 1
14.3 Listing the Boot Inventory-SourceSetting Class

Each Boot Configuration Representation contains an ordered list of boot sources, which indicate the logical devices to use during the boot process.

Enumerate the **BootSourceSetting** class with the following parameters and syntax:

**EXAMPLE:**

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BootSourceSetting
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

```
DCIM_BootSourceSetting
  BIOSBootString = Embedded SATA Port A Optical: SATA Optical Drive BootSeq
  BootString = Embedded SATA Port A Optical: SATA Optical Drive BootSeq
  CurrentAssignedSequence = 0
  CurrentEnabledStatus = 1
  ElementName = Embedded SATA Port A Optical: SATA Optical Drive BootSeq
  FailThroughSupported = 1
  InstanceID = IPL:Optical.SATAEmbedded.A-1:eb8aeb15796fb85f8e1447f0cfc8a68e
  PendingAssignedSequence = 0
  PendingEnabledStatus = 1

DCIM_BootSourceSetting
  BIOSBootString = Hard drive C: BootSeq
  BootString = Hard drive C: BootSeq
  CurrentAssignedSequence = 1
  CurrentEnabledStatus = 1
  ElementName = Hard drive C: BootSeq
  FailThroughSupported = 1
  InstanceID = IPL:HardDisk.List.1-1:c9203080df84781e2ca3d512883dee6f
  PendingAssignedSequence = 1
  PendingEnabledStatus = 1
```

The **ChangeBootOrderByInstanceID** method in Section 14.4 will use the **InstanceID** field as input.

14.4 Changing the Boot Order by InstanceID-ChangeBootOrderByInstanceID()

The **ChangeBootOrderByInstanceID()** method is called to change the boot order of boot sources within a configuration. The method’s input parameter, **source**, is an ordered array of **InstanceIDs** of **BootSourceSetting** instances.
The *CurrentAssignedSequence* attribute of each instance, from Section 14.3, defines the instance’s place in the zero based indexed boot sequence. Note: In order for the changes to be applied, the `CreateTargetedConfigJob()` method in Section 17.7 must be executed.

Invoke `ChangeBootOrderByInstanceID()` with the following parameters and syntax:

```
[INSTANCE ID]: Obtained from the BootSourceSetting Class enumeration, this example uses the field IPL
```

**source:** Reference to the `InstanceID` attribute from Section 14.3

**EXAMPLE:**

```
winrm i ChangeBootOrderByInstanceID http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BootConfigSetting
?InstanceId=[INSTANCE ID]
-u:[USER] -p:[PASSWORD]
-r://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:ChangeBootOrderByInstanceID.xml
```

The `source` input is obtained from the `BootSourceSetting` inventory in Section 14.3

The input file `ChangeBootOrderByInstanceID.xml` is shown below:

```
</p:ChangeBootOrderByInstanceID_INPUT>
```

**OUTPUT:**

```
ChangeBootOrderByInstanceID_OUTPUT
Message = The command was successful
MessageID = BOOT001
ReturnValue = 0
```

### 14.5 Enable or Disable the Boot Source-ChangeBootSourceState()

The `ChangeBootSourceState()` method is called to change the enabled status of `BootSourceSetting` instances to `Disable` or `Enable`. The input parameter, `source`, is an array of `InstanceId` of `BootSourceSetting` instances. Enumerating the `BootSourceSetting` Class in Section 14.3, displays the `CurrentEnabledStatus` field which provides the applicable status.

Note 1: In order for the changes to be applied, the `CreateTargetedConfigJob()` method in Section 17.7 must be executed.

Note 2: BIOS does not support the setting of `EnabledState` for BCV devices.

Invoke `ChangeBootSourceState()` with the following parameters and syntax:
[INSTANCE ID]: Obtained from the BootSourceSetting Class enumeration, this example uses the field IPL

source: Reference to the InstanceID attribute from Section 14.3

EnabledState: State of boot source element

Disabled=0, Enabled=1

EXAMPLE:

winrm i ChangeBootSourceState http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BootConfigSetting
?InstanceID=[INSTANCE ID]
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:ChangeBootSourceState.xml

The input file ChangeBootSourceState.xml is shown below:

```xml
    <p:EnabledState>0</p:EnabledState>
    <p:source>IPL:Optical.SATAEmbedded.A-1:eb8aeb15796fb85f8e1447f0cfb8a68e</p:source>
</p:ChangeBootSourceState_INPUT>
```

OUTPUT:

ChangeBootSourceState_OUTPUT
Message = The command was successful
MessageID = BOOT001
ReturnValue = 0

15 NIC/CNA Card Management

This feature provides the ability to get and set the Network Interface (NIC) Card or Converged Network Adapter (CNA) attributes that are configurable using NIC/CNA Option-ROM or NIC/CNA UEFI HII. The attributes include functionalities for the following:

- Partition and personality (CNA only)
- iSCSI boot and PXE boot that are part of the NIC/CNA firmware

The ability to configure CNAs has been added to the NIC profile that extends the management capabilities of the referencing profiles. The NICs/CNAs are modeled as views with collections of attributes where there is a view for each partition on the controller.

The NIC/CNA Inventory has these classes and views:

1. DCIM_NICEnumeration, (see Section 15.1)
2. DCIM_NICString (see Section 15.2)
3. DCIM_NICInteger (see Section 15.3)
4. DCIM_NICView (see Section 15.4)
5. DCIM_NICCapabilities (see Section 15.5)
6. DCIM_NICStatistics (see Section 15.6)

Profile and Associated MOFS:
http://www.delltechcenter.com/page/DCIM.Library.Profile

15.1 Listing the NIC/CNA Inventory-Enumeration Class

Enumerate the NICEnumeration class with the following parameters and syntax:

EXAMPLE - CNA:

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICEnumeration
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

OUTPUT - CNA: For SAMPLE PORT 1 / PARTITION 1 (all attributes on all partitions are enumerated)

```
DCIM_NICEnumeration
   AttributeDisplayName = TCP/IP Parameters via DHCP
   AttributeName = TcpIpViaDHCP
   CurrentValue = Enabled
   Dependency = <Dep><AttrLev Op="OR"><ROIf Name="IpVer">IPv6</ROIf><ROIf Name="iSCSIBootSupport">Unavailable</ROIf></AttrLev></Dep>
   FQDD = NIC.Integrated.1-1-1
   GroupDisplayName = iSCSI General Parameters
   GroupID = IscsiGenParams
   InstanceID = NIC.Integrated.1-1-1:TcplpViaDHCP
   IsReadOnly = false
   PendingValue = null
   PossibleValues = Disabled, Enabled
   PossibleValuesDescription = Disabled, Enabled

DCIM_NICEnumeration
   AttributeDisplayName = IP Autoconfiguration
   AttributeName = IpAutoConfig
   CurrentValue = Enabled
   Dependency = <Dep><AttrLev Op="OR"><ROIf Name="IpVer">IPv4</ROIf><ROIf Name="iSCSIBootSupport">Unavailable</ROIf></AttrLev></Dep>
   FQDD = NIC.Integrated.1-1-1
   GroupDisplayName = iSCSI General Parameters
   GroupID = IscsiGenParams
   InstanceID = NIC.Integrated.1-1-1:IpAutoConfig
   IsReadOnly = true
   PendingValue = null
```
PossibleValues = Disabled, Enabled
PossibleValuesDescription = Disabled, Enabled

DCIM_NICEnumeration
  AttributeDisplayName = iSCSI Parameters via DHCP
  AttributeName = IscsiViaDHCP
  CurrentValue = Enabled
  Dependency = <Dep><AttrLev Op="OR"><ROIf Name="iSCSIBootSupport">Unavailable</ROIf></AttrLev></Dep>
  FQDD = NIC.Integrated.1-1-1
  GroupDisplayName = iSCSI General Parameters
  GroupID = IscsiGenParams
  InstanceID = NIC.Integrated.1-1-1:IscsiViaDHCP
  IsReadOnly = false
  PendingValue = null
  PossibleValues = Disabled, Enabled
  PossibleValuesDescription = Disabled, Enabled

DCIM_NICEnumeration
  AttributeDisplayName = CHAP Authentication
  AttributeName = ChapAuthEnable
  CurrentValue = Disabled
  Dependency = <Dep><AttrLev Op="OR"><ROIf Name="iSCSIBootSupport">Unavailable</ROIf></AttrLev></Dep>
  FQDD = NIC.Integrated.1-1-1
  GroupDisplayName = iSCSI General Parameters
  GroupID = IscsiGenParams
  InstanceID = NIC.Integrated.1-1-1:ChapAuthEnable
  IsReadOnly = false
  PendingValue = null
  PossibleValues = Disabled, Enabled
  PossibleValuesDescription = Disabled, Enabled

15.2  Listing the NIC/CNA Inventory-String Class

Enumerate DCIM_NICString class with the following parameters and syntax:

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICString
-u:[USER] -p:[PASSWORD]
-f:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic

OUTPUT:

DCIM_NICString
  AttributeDisplayName = Chip Type
  AttributeName = ChipMdl
  CurrentValue = BCM5720 A0
  Dependency = null
  FQDD = NIC.Integrated.1-1-1
GroupDisplayName = Broadcom Main Configuration Page
GroupId = VndrConfigPage
InstanceID = NIC.Integrated.1-1-1:ChipMdl
IsReadOnly = true
MaxLength = 0
MinLength = 0
PendingValue = null
ValueExpression = null

DCIM_NICString
AttributeDisplayName = PCI Device ID
AttributeName = PCIDeviceID
CurrentValue = 165F
Dependency = null
FQDD = NIC.Integrated.1-1-1
GroupDisplayName = Broadcom Main Configuration Page
GroupId = VndrConfigPage
InstanceID = NIC.Integrated.1-1-1:PCIDeviceID
IsReadOnly = true
MaxLength = 0
MinLength = 0
PendingValue = null
ValueExpression = null

DCIM_NICString
AttributeDisplayName = Bus:Dev:Func
AttributeName = BusDeviceFunction
CurrentValue = 01:00:00
Dependency = null
FQDD = NIC.Integrated.1-1-1
GroupDisplayName = Broadcom Main Configuration Page
GroupId = VndrConfigPage
InstanceID = NIC.Integrated.1-1-1:BusDeviceFunction
IsReadOnly = true
MaxLength = 0
MinLength = 0
PendingValue = null
ValueExpression = null

DCIM_NICString
AttributeDisplayName = Link Status
AttributeName = LinkStatus
CurrentValue = UP
Dependency = null
FQDD = NIC.Integrated.1-1-1
GroupDisplayName = Broadcom Main Configuration Page
GroupId = VndrConfigPage
InstanceID = NIC.Integrated.1-1-1:LinkStatus
IsReadOnly = true
MaxLength = 0
MinLength = 0
PendingValue = null
ValueExpression = null
15.3 Listing the CNA Inventory-Integer Class

Enumerate the DCIM_NICInteger class with the following parameters and syntax:

**EXAMPLE**:

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICInteger
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT**:

```
DCIM_NICInteger
   AttributeDisplayName = Blink LEDs
   AttributeName = BlnkLeds
   CurrentValue = 0
   Dependency = null
   FQDD = NIC.Integrated.1-1-1
   GroupDisplayName = Broadcom Main Configuration Page
   GroupID = VndrConfigPage
   InstanceID = NIC.Integrated.1-1-1:BlnkLeds
   IsReadOnly = false
   LowerBound = 0
   PendingValue = null
   UpperBound = 15

DCIM_NICInteger
   AttributeDisplayName = Link Up Delay Time
   AttributeName = LnkUpDelayTime
   CurrentValue = 0
   Dependency = <Dep><AttrLev Op="OR"><ROIf Name="iSCSIBootSupport">Unavailable</ROIf></AttrLev></Dep>
   FQDD = NIC.Integrated.1-1-1
   GroupDisplayName = iSCSI General Parameters
   GroupID = IscsiGenParams
   InstanceID = NIC.Integrated.1-1-1:LnkUpDelayTime
   IsReadOnly = false
   LowerBound = 0
   PendingValue = null
   UpperBound = 255

DCIM_NICInteger
   AttributeDisplayName = LUN Busy Retry Count
   AttributeName = LunBusyRetryCnt
   CurrentValue = 0
   Dependency = <Dep><AttrLev Op="OR"><ROIf Name="iSCSIBootSupport">Unavailable</ROIf></AttrLev></Dep>
   FQDD = NIC.Integrated.1-1-1
```
GroupDisplayName = iSCSI General Parameters
GroupId = IscsiGenParams
InstanceId = NIC.Integrated.1-1-1:LunBusyRetryCnt
IsReadOnly = false
LowerBound = 0
PendingValue = null
UpperBound = 60

DCIM_NICInteger
  AttributeDisplayName = TCP Port
  AttributeName = FirstTgtTcpPort
  CurrentValue = 3260
  Dependency = <Dep><AttrLev Op="OR"><ROIf Name="iSCSIBootSupport">Unavailable</ROIf></AttrLev></Dep>
  FQDD = NIC.Integrated.1-1-1
  GroupDisplayName = iSCSI First Target Parameters
  GroupId = IscsiFirstTgtParams
  InstanceId = NIC.Integrated.1-1-1:FirstTgtTcpPort
  IsReadOnly = false
  LowerBound = 1
  PendingValue = null
  UpperBound = 65535

15.4 Listing the CNA Inventory-NICView Class

Enumerate the DCIM_NICView class with the following parameters and syntax:

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICView
-u:[USER] -p:[PASSWORD]
-r://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic

OUTPUT FOR FIRST and SECOND PORT (NICView will return all ports and partitions):

DCIM_NICView
  AutoNegotiation = 0
  BusNumber = 1
  ControllerBIOSVersion = 1.17
  CurrentMACAddress = 14:FE:B5:FF:B3:EA
  DataBusWidth = 0002
  DeviceNumber = 0
  EFIVersion = 15.0.16
  FCoEOffloadMode = 3
  FCoEVWNW = null
  FQDD = NIC.Integrated.1-1-1
  FamilyVersion = 7.0.39
  FunctionNumber = 0
InstanceId = NIC.Integrated.1-1-1
LastSystemInventoryTime = 20010708151620.000000+000
LastUpdateTime = 20010708151606.000000+000
LinkDuplex = 0
LinkSpeed = 0
MaxBandwidth = 0
MediaType = 4
MinBandwidth = 0
NicMode = 3
PCIDeviceID = 165f
PCISubDeviceID = 1f5b
PCISubVendorID = 1028
PCIVendorID = 14e4
PermanentFCOEMACAddress
PermanentMACAddress = 14:FE:B5:FF:B3:EA
PermanentiSCSIMACAddress
ProductName = Broadcom Gigabit Ethernet BCM5720 - 14:FE:B5:FF:B3:EA
ReceiveFlowControl = 0
SlotLength = 0002
SlotType = 0002
TransmitFlowControl = 0
VendorName = null
WWPN = null
iScsiOffloadMode = 3
DCIM_NICView
AutoNegotiation = 0
BusNumber = 1
ControllerBIOSVersion = 1.17
CurrentMACAddress = 14:FE:B5:FF:B3:EB
DataBusWidth = 0002
DeviceNumber = 0
EFIVersion = 15.0.16
FCoEOffloadMode = 3
FCoEWWNN = null
FQDD = NIC.Integrated.1-2-1
FamilyVersion = 7.0.39
FunctionNumber = 1
InstanceId = NIC.Integrated.1-2-1
LastSystemInventoryTime = 20010708151620.000000+000
LastUpdateTime = 20010708151606.000000+000
LinkDuplex = 0
LinkSpeed = 0
MaxBandwidth = 0
MediaType = 4
MinBandwidth = 0
NicMode = 3
PCIDeviceID = 165f
PCISubDeviceID = 1f5b
PCISubVendorID = 1028
PCIVendorID = 14e4
PermanentFCOEMACAddress
PermanentMACAddress = 14:FE:B5:FF:B3:EB
PermanentiSCSIMACAddress
ProductName = Broadcom Gigabit Ethernet BCM5720 - 14:FE:B5:FF:B3:EB
ReceiveFlowControl = 0
SlotLength = 0002
SlotType = 0002
TransmitFlowControl = 0
VendorName = null
WWPN = null
iScsiOffloadMode = 3

15.5 Listing the CNA Inventory-NICCapabilities Class

Enumerate the DCIM_NICCapabilities class with the following parameters and syntax:

**EXAMPLE:**

```
winrm http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICCapabilities
-u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

```
DCIM_NICCapabilities
  BPESupport = 3
  CongestionNotification = 3
  DCBExchangeProtocol = 3
  ETS = 3
  EVBModesSupport = 3
  EnergyEfficientEthernet = 2
  FCoEBootSupport = 3
  FCoEMaxIOsPerSession = 0
  FCoEMaxNPIVPerPort = 0
  FCoEMaxNumberExchanges = 0
  FCoEMaxNumberLogins = 0
  FCoEMaxNumberOffCTargets = 0
  FCoEMaxNumberOfFCTargets = 0
  FCoEOffloadSupport = 3
  FQDD = NIC.Integrated.1-1-1
  FeatureLicensingSupport = 3
  FlexAddressingSupport = 2
  IPSecOffloadSupport = 3
  InstanceID = NIC.Integrated.1-1-1
  MACSecSupport = 3
  NWManagementPassThrough = 2
  NicPartitioningSupport = 3
  OSBMCManagementPassThrough = 2
  OnChipThermalSensor = 2
  OpenFlowSupport = 3
  PXEBootSupport = 2
  PartitionWOLSupport = 3
  PriorityFlowControl = 3
  RDMASupport = 3
  RXFlowControl = 3
```
RemotePHY = 3
TCPChimneySupport = 3
TXBandwidthControlMaximum = 3
TXBandwidthControlMinimum = 3
TXFlowControl = 3
VEBVEPAMultiChannel = 3
VEBVEPASingleChannel = 3
VFSRIOVSupport = 3
VirtualLinkControl = 3
WOLSupport = 2
iSCSIBootSupport = 2
iSCSIOffloadSupport = 3
uEFISupport = 2

15.6 Listing the CNA Inventory- NICStatistics Class

Enumerate the DCIM_NICStatistics class with the following parameters and syntax:

EXAMPLE:

```
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

OUTPUT:

```
DCIM_NICStatistics
 DiscardedPkts = 0
 FCCRCErrorCount = null
 FCOELinkFailures = null
 FCOEPktRxCount = null
 FCOEPktTxCount = null
 FCOERxPktDroppedCount = null
 FQDD = NIC.Integrated.1-1-1
 InstanceID = NIC.Integrated.1-1-1
 LinkStatus = 1
 OSDriverState = 1
 PartitionLinkStatus = null
 PartitionOSDriverState = null
 RxBroadcast = 65177
 RxBytes = null
 RxErrorPktAlignmentErrors = 0
 RxErrorPktFCSErrors = 0
 RxFalseCarrierDetection = null
 RxJabberPkt = null
 RxMulticast = 11000
 RxPauseXOFFFrames = 0
 RxPauseXONFrames = 0
 RxRuntPkt = null
 RxUnicast = 0
 StartStatisticTime = 20111208013952.000000+000
 StatisticTime = 20111208073904.000000+000
```
Applying the Pending Values for CNA-
CreateTargetedConfigJob()

The CreateTargetedConfigJob() method is called to apply the pending values created using the SetAttribute() and SetAttributes() methods. The system automatically reboots depending on the ScheduledStartTime selected. Use the CreateTargetedConfigJob() jobId output to get the status (see Section 10.0).

Invoke CreateTargetedConfigJob() with the following parameters and syntax:

- **Target**: This parameter is the FQDD, which is found by enumerating the CNA attributes in Section 15.1.

- **RebootJobType**: There are three options for rebooting the system.
  1. PowerCycle
  2. Graceful Reboot without forced shutdown
  3. Graceful reboot with forced shutdown

  Note: When a user does not want to set a reboot type while creating a target job, users should comment out the RebootJobType in the input xml. User should not enter “0” or give no parameter in the input xml.

- **ScheduledStartTime & UntilTime**: See Section 3.2.4

**EXAMPLE:**

```
winrm i CreateTargetedConfigJob http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_NICService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:NICService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck -encoding=utf-8 -a:basic -
file:CreateTargetedConfigJob_CNA.xml
```

The input file CreateTargetedConfigJob_CNA.xml is shown below:

```
```

134
<p:Target>NIC.Integrated.1-1-1</p:Target>
<p:RebootJobType>1</p:RebootJobType>
<p:ScheduledStartTime>TIME_NOW</p:ScheduledStartTime>
<p:UntilTime>20201111111111</p:UntilTime>
</p:CreateTargetedConfigJob_INPUT>

**OUTPUT:**

When this method is executed, a *jobid* or an error message is returned. The status of this *jobid* can be checked within the job control provider in Section 10.

CreateTargetedConfigJob_OUTPUT

Job

Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
SelectorSet
  Selector: InstanceID = JID_001269609760, __cimnamespace = root/dcim
ReturnValue = 4096

15.8 Deleting the Pending Values for CNA-DeletePendingConfiguration()

The **DeletePendingConfiguration()** method cancels the pending configuration changes made before the configuration job is created using the **CreateTargetedConfigJob()** method. This method only operates on the pending changes before running the **CreateTargetedConfigJob()** method. After the configuration job is created, to cancel the pending changes, call the **DeleteJobQueue()** method in the Job Control profile.

Invoke the **DeletePendingConfiguration()** method with the following parameters and syntax:

**EXAMPLE:**

```
winrm i DeletePendingConfiguration http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_NICService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:NICService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -
file:DeletePendingConfiguration_CNA.xml
```

The input file **DeletePendingConfiguration_CNA.xml** is shown below:

```
  <p:Target>NIC.Integrated.1-1-1</p:Target>
</p:DeletePendingConfiguration_INPUT>
```
OUTPUT:

DeletePendingConfiguration_OUTPUT
    Message = The command was successful
    MessageID = NIC001
    ReturnValue = 0

15.9 Getting the CNA Enumeration Instance

Use the following example to get an instance of the DCIM_NICEnumeration class.

Get a DCIM_NICEnumeration class instance from the first port and first partition with the following parameters and syntax:

    [INSTANCEID]: This is obtained from the enumeration in Section 15.1, in which this example would use NIC.Integrated.1-1-1 as an InstanceID.

EXAMPLE:

winrm g http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICEnumeration
?InstanceID=[INSTANCEID]
-r:https://[IPADDRESS]:443/wsman
-u:[USER] -p:[PASSWORD]
-auth:basic -encoding:utf-8 -SkipCNCheck -SkipCACheck

OUTPUT:

DCIM_NICEnumeration
    AttributeDisplayName = iSCSI Offload Mode
    AttributeName = iScsiOffloadMode
    CurrentValue = Disabled
    Dependency = <Dep><AttrLev Op="OR"><ROIf Name="NicMode\133Partition\0723\135">Enabled</ROIf></AttrLev></Dep>
    FQDD = NIC.Integrated.1-1-3
    GroupDisplayName = PARTITION 3 CONFIGURATION
    GroupID = ConfigureForm3
    InstanceID = NIC.Integrated.1-1-3:iScsiOffloadMode
    IsReadOnly = false
    PendingValue = null
    PossibleValues = Disabled, Enabled
    PossibleValuesDescription = Disabled, Enabled

15.10 Setting the IscsiOffloadMode Attribute

The SetAttribute() method is used to set or change the value of a CNA attribute. Enable the NICMode, IscsiOffloadMode, and FcoeOffloadMode personality attributes to enable the corresponding personalities: NIC, iSCSI, and FCOE.
For Broadcom CNA cards, the partitions on each port can be set to any personality. NICMode can always be enabled or disabled for any of the given partitions. For the IscsiOffloadMode and FcoeOffloadMode personalities, up to two personalities can be enabled on each port.

For the Qlogic CNA cards, partition three can be set to either NICMode or IscsiOffloadMode. Partition four can be set to either NICMode or FcoeOffloadMode.

Invoke the SetAttribute() method with the following parameters (from Section 15.1) and syntax:

- **Target:** FQDD attained through DCIM_NICEnumeration
- **AttributeName:** Attained from AttributeName field
- **AttributeValue:** A new value to assign to the specified NICAttribute. If this value is valid, it is applied to the PendingValue property or the Currentvalue property of the specified NICAttribute. Possible choices are attained from PossibleValues field, such as:
  - Possible values: Disabled, Enabled

**EXAMPLE:**

```bash
winrm i SetAttribute http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_NICService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:NICService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:SetAttribute_NIC.xml
```

The information in the input file SetAttribute_NIC.xml is shown below:

```xml
<p:Target>NIC.Integrated.1-1-1</p:Target>
<p:AttributeName>IscsiOffloadMode</p:AttributeName>
<p:AttributeValue>Enabled</p:AttributeValue>
</p:SetAttributes_INPUT>
```

**OUTPUT:**

SetAttribute_OUTPUT
  - Message = The command was successful
  - MessageID = NIC001
  - RebootRequired = Yes
  - ReturnValue = 0
  - SetResult = Set PendingValue
15.11 Setting the MaxBandwidth Attribute

The `SetAttribute()` method is used to set or change the value of a CNA attribute.

The MinBandwidth and MaxBandwidth attributes control the bandwidth allocations for a given CNA partition. The values are displayed in percentage.

For Broadcom CNA cards, the MinBandwidth attribute values for a given port must always add up to either 0 or 100. MaxBandwidth is a value of 100 or less for any given partition.

For the Qlogic CNA cards, the MinBandwidth attribute values for a given port must add up to 100 or less. MaxBandwidth again is a value of 100 or less for any given partition.

Invoke `SetAttribute()` with the following parameters (from Section 15.1) and syntax:

**Target:** FQDD attained through `DCIM_NICInterface`

**AttributeName:** Attained from `AttributeName` field

**AttributeValue:** A new value to assign to the specified `NICAttribute`. If this value is valid, it is applied to the `PendingValue` property or the `CurrentValue` property of the specified `NICAttribute`. Range of choices is attained from the `LowerBound` and `UpperBound` fields:

- `LowerBound = 0`
- `UpperBound = 100`

**EXAMPLE:**

```
winrm i SetAttribute http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_NICService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:NICService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:SetAttribute_NIC.xml
```

The input file `SetAttribute_NIC.xml` is shown below:

```xml
<p:Target>NIC.Integrated.1-1-2</p:Target>  
<p:AttributeName>MaxBandwidth</p:AttributeName>  
<p:AttributeValue>75</p:AttributeValue> 
</p:SetAttributes_INPUT>
```

**OUTPUT:**

```
SetAttribute_OUTPUT  
Message = The command was successful
```
MessageID = NIC001
RebootRequired = Yes
ReturnValue = 0
SetResult = Set PendingValue

15.12 Setting the VirtMacAddr Attribute

The SetAttribute() method is used to set or change the value of a CNA attribute. The I/O identity string attributes: (VirtMacAddr, VirtIscsiMacAddr, VirtFIPMacAddr, VirtWWN, and VirtWWPN) display a unique behavior. After setting them to a non-default value, the attribute values are retained until there is AC power supply. If the AC power supply is disconnected, the attributes revert to their default values.

Invoke the SetAttribute() method with the following parameters and syntax:

**Target**: FQDD attained through `DCIM_NICString`

`AttributeName`: Attained from `AttributeName` field

`AttributeValue`: A new value to assign to the specified NICAttribute. If this value is valid, it is applied to the `PendingValue` property or the `CurrentValue` property of the specified NICAttribute. The range of acceptable strings is present in the `MinLength` and `MaxLength` fields.

**EXAMPLE**:  
winrmi SetAttribute http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_NICService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:NICService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:SetAttribute_NIC.xml

The input file `SetAttribute_NIC.xml` is shown below:

```xml
  <p:Target>NIC.Integrated.1-1-2</p:Target>
  <p:AttributeName>VirtMacAddr</p:AttributeName>
</p:SetAttributes_INPUT>
```

**OUTPUT:**

```
SetAttribute_OUTPUT
  Message = The command was successful
  MessageID = NIC001
```
RebootRequired = Yes
ReturnValue = 0
SetResult = Set PendingValue

15.13 Setting the *LegacyBootProto* Attribute

The `SetAttribute()` method is used to set or change the value of a NIC attribute.

**WARNING:** The local BIOS setting always overwrites the *LegacyBootProto* option. This option is only applied in the BIOS setup. By setting this attribute remotely, it appears that the value is set, but it really did not because the local BIOS setting overrides it. Running a ‘get’ on the attribute remotely displays a different current value.

Invoke `SetAttribute()` with the following parameters(from Section 15.1) and syntax:

**Target:** FQDD attained through *DCIM_NICEnumeration*

**AttributeName:** Attained from `AttributeName` field

**AttributeValue:** A new value to assign to the specified `NICAttribute`. If this value is valid, it will be applied to the `PendingValue` property or the `CurrentValue` property of the specified `NICAttribute`. Possible choices are attained from `PossibleValues` field, such as:

Possible values: PXE, iSCSI, NONE

**EXAMPLE:**

```
winrm i SetAttribute http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_NICService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:NICService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:SetAttribute_NIC.xml
```

The input file `SetAttribute_NIC.xml` is shown below:

```
  <p:Target>NIC.Embedded.1-1</p:Target>
  <p:AttributeName>LegacyBootProto</p:AttributeName>
  <p:AttributeValue>PXE</p:AttributeValue>
</p:SetAttributes_INPUT>
```

**OUTPUT:**

```
SetAttribute_OUTPUT
  Message = The command was successful
  MessageID = NIC001
```
RebootRequired = Yes
ReturnValue = 0
SetResult = Set PendingValue

15.14 Setting CNA LAN Modes
The SetAttributes() method is used to set or change the values of a group of NIC attributes.

Invoke SetAttributes() with the following parameters (from Section 15.1) and syntax:

Target:  FQDD attained through DCIM_NICEnumeration

AttributeName:  Attained from AttributeName field

AttributeValue:  A new value to assign to the specified NICAttribute. If this value is valid, it will be applied to the PendingValue property or the CurrentValue property of the specified NICAttribute. Possible choices are attained from PossibleValues field.

EXAMPLE:

winrm i SetAttributes http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_NICService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:NICService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:SetAttributes_NIC.xml

The input file SetAttributes_NIC.xml is shown below:

```xml
  <p:Target>NIC.1-1</p:Target>
  <p:AttributeName>LegacyBootProto</p:AttributeName>
  <p:AttributeValue>PXE</p:AttributeValue>
  <p:AttributeName>LknSpeed</p:AttributeName>
  <p:AttributeValue>10Mbps Half</p:AttributeValue>
  <p:AttributeName>WakeOnLan</p:AttributeName>
  <p:AttributeValue>Disabled</p:AttributeValue>
  <p:AttributeName>VLanMode</p:AttributeName>
  <p:AttributeValue>Enabled</p:AttributeValue>
  <p:AttributeName>IscsiTgtBoot</p:AttributeName>
  <p:AttributeValue>One Time Disabled</p:AttributeValue>
</p:SetAttributes_INPUT>
```

OUTPUT:

SetAttributes_OUTPUT
  Message = The command was successful
  MessageID = NIC001
15.15 Setting the iSCSI Boot Target

The `SetAttributes()` method is used to set or change the values of the iSCSI boot target attributes.

Invoke the `SetAttributes()` method with the following parameters (from 15.1) and syntax:

- **Target**: FQDD attained through `DCIM_NICEnumeration`
- **AttributeName**: Attained from `AttributeName` field
- **AttributeValue**: A new value to assign to the specified `NICAttribute`. If this value is valid, it is applied to the `PendingValue` property or the `CurrentValue` property of the specified `NICAttribute`. Possible choices are attained from `PossibleValues` field, such as:
  - Possible values: Disabled, Enabled

**EXAMPLE**:

```shell
winrm i SetAttributes http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_NICService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:NICService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:SetAttributes_iSCSI_BootTarget.xml
```

The information in the input file `SetAttributes_iSCSI_BootTarget.xml` is shown below:

```xml
  <p:Target>NIC.Integrated.1-1-1</p:Target>
  <p:AttributeName>BootToTarget</p:AttributeName>
  <p:AttributeValue>Enabled</p:AttributeValue>
  <p:AttributeName>IscsiInitiatorIpAddr</p:AttributeName>
  <p:AttributeValue>10.10.10.1</p:AttributeValue>
  <p:AttributeName>IscsiInitiatorSubnet</p:AttributeName>
  <p:AttributeValue>255.255.255.0</p:AttributeValue>
  <p:AttributeName>IscsiInitiatorGateway</p:AttributeName>
  <p:AttributeValue>10.10.10.1</p:AttributeValue>
  <p:AttributeName>IscsiInitiatorPrimDns</p:AttributeName>
  <p:AttributeValue>10.10.10.2</p:AttributeValue>
  <p:AttributeName>IscsiInitiatorSecDns</p:AttributeName>
  <p:AttributeValue>10.10.10.3</p:AttributeValue>
  <p:AttributeName>IscsiInitiatorName</p:AttributeName>
</p:SetAttributes_INPUT>
<p:AttributeValue>testname</p:AttributeValue>
<p:AttributeName>lsclsiInitiatorChapId</p:AttributeName>
<p:AttributeValue>testid</p:AttributeValue>
<p:AttributeName>lsclsiInitiatorChapPwd</p:AttributeName>
<p:AttributeValue>testpassword</p:AttributeValue>
<p:AttributeName>FirstTgtIpAddress</p:AttributeName>
<p:AttributeValue>2.2.2.2</p:AttributeValue>
<p:AttributeName>FirstTgtIscsiName</p:AttributeName>
<p:AttributeValue>tgtiscsitest</p:AttributeValue>
<p:AttributeName>FirstTgtChapId</p:AttributeName>
<p:AttributeValue>firsttestID</p:AttributeValue>
<p:AttributeValue>firsttestpwd</p:AttributeValue>
<p:AttributeValue>testpassword2</p:AttributeValue>
</p:SetAttributes_INPUT>

**OUTPUT:**

SetAttribute_OUTPUT

- Message = The command was successful
- MessageID = NIC001
- RebootRequired = Yes
- ReturnValue = 0
- SetResult = Set PendingValue

### 15.16 Setting the FCoE Boot Target

The **SetAttributes()** method is used to set or change the values of the FCoE boot target attributes.

Invoke the **SetAttributes()** method with the following parameters (from 15.1) and syntax:

**Target:** FQDD attained through *DCIM_NICEnumeration*

**AttributeName:** Attained from *AttributeName* field

**AttributeValue:** A new value to assign to the specified *NICAttribute*. If this value is valid, it is applied to the *PendingValue* property or the *CurrentValue* property of the specified *NICAttribute*. Possible choices are attained from *PossibleValues* field, such as:

Possible values: Disabled, Enabled

**EXAMPLE:**

```
winrm i SetAttributes http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_NICService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_NICService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:NICService
-u:[USER] -p:[PASSWORD]
```
The information in the input file `SetAttributes_FCoE_BootTarget.xml` is shown below:

```xml
  <p:Target>NIC.Integrated.1-1-1</p:Target>
  <p:AttributeName>ConnectFirstFCoETarget</p:AttributeName>
  <p:AttributeValue>Enabled</p:AttributeValue>
  <p:AttributeName>FirstFCoEWWPNTarget</p:AttributeName>
  <p:AttributeValue>20:00:00:10:18:88:C0:03</p:AttributeValue>
  <p:AttributeName>FirstFCoEBootTargetLUN</p:AttributeName>
  <p:AttributeValue>33</p:AttributeValue>
  <p:AttributeName>FirstFCoEFCFVLANID</p:AttributeName>
  <p:AttributeValue>34</p:AttributeValue>
</p:SetAttributes_INPUT>
```

**OUTPUT:**

```
SetAttribute_OUTPUT
  Message = The command was successful
  MessageID = NIC001
  RebootRequired = Yes
  ReturnValue = 0
  SetResult = Set PendingValue
```

## 16 RAID Storage Management

The remote RAID configuration allows users to remotely query and configure the Hardware RAID of the system. The RAID profile extends the management capabilities of referencing profiles by adding the capability to represent the configuration of RAID storage. The RAID storage is modeled as collections of attributes where there are collections for the storage adaptors, physical disks, logical disks, end enclosures and parent-child relationships between the collections. Additionally, there is a configuration service that contains all the methods used to configure the RAID storage.

**Profile and Associated MOFs:**


The RAID Inventory contains the following attributes:

- **DCIM_RAIDE Enumeration** (16.1)
- **DCIM_RAIDInteger** (16.3)
- **DCIM_RAIDString** (16.5)
- **DCIM_ControllerView** (16.7)
- **DCIM_PhysicalDiskView** (16.9)
16.1 Listing the RAID Inventory-Enumeration Class

The RAID Inventory has these attributes: DCIM_RAIDEnumeration (this section), DCIM_RAIDInteger (Section 16.3), and DCIM_RAIDString (see Section 16.5).

Enumerate the DCIM_RAIDEnumeration class to display all the RAID controllers and virtual disk attributes in a system.

Enumerate the DCIM_RAIDEnumeration class with the following parameters and syntax:

EXAMPLE:

```plaintext
```

OUTPUT:

```
DCIM_RAIDEnumeration
     AttributeName = RAIDSupportedDiskProt
     CurrentValue = SAS, SATA
     FQDD = RAID.Integrated.1-1
     InstanceID = RAID.Integrated.1-1:RAIDSupportedDiskProt
     IsReadOnly = true
     PendingValue
     PossibleValues = SAS, SATA

DCIM_RAIDEnumeration
     AttributeName = RAIDloadBalancedMode
     CurrentValue = Automatic
     FQDD = RAID.Integrated.1-1
     InstanceID = RAID.Integrated.1-1:RAIDloadBalancedMode
     IsReadOnly = false
     PendingValue
     PossibleValues = Automatic, Disabled

DCIM_RAIDEnumeration
     AttributeName = RAIDBatteryLearnMode
     CurrentValue = Automatic
     FQDD = RAID.Integrated.1-1
     InstanceID = RAID.Integrated.1-1:RAIDBatteryLearnMode
     IsReadOnly = false
     PendingValue
     PossibleValues = Automatic, Warn only, Disabled

DCIM_RAIDEnumeration
     AttributeName = RAIDdefaultWritePolicy
```

The ‘get’ instance method in section 16.2 uses this `InstanceID` as input.

The ‘set attribute’ method in section 16.19.1 uses the `FQDD`, `AttributeName`, and `PossibleValues` fields as input.

The ‘set attributes’ method in section 16.19.2 uses the `FQDD`, `AttributeName`, and `PossibleValues` fields as input.
CurrentValue = WriteBack  
FQDD = Disk.Virtual.1:RAID.Integrated.1-1
InstanceID = Disk.Virtual.1:RAID.Integrated.1-1:RAIDdefaultWritePolicy  
IsReadOnly = false  
PendingValue  
PossibleValues = WriteThrough, WriteBack, WriteBackForce

### 16.2 Getting a RAID Enumeration Instance

Use the following example to get an instance of the DCIM_RAIDEnumeration class instead of all the instances as shown in Section 16.1.

Get a RAIDEnumeration instance with the following parameters and syntax:

```
[INSTANCEID]: This is obtained from the enumeration in Section 16.1, which shows an example using RAID.Integrated.1-1:RAIDloadBalancedMode as an instanceID.
```

**EXAMPLE**:

```
winrm g cimv2/root/dcim/DCIM_RAIDEnumeration?InstanceID=[[INSTANCE ID]]
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT**:

DCIM_RAIDEnumeration  
AttributeName = RAIDloadBalancedMode  
CurrentValue = Automatic  
FQDD = RAID.Integrated.1-1  
InstanceID = RAID.Integrated.1-1:RAIDloadBalancedMode  
IsReadOnly = false  
PendingValue  
PossibleValues = Automatic, Disabled

### 16.3 Listing the RAID Inventory-Integer Class

The RAID Inventory has these attributes: DCIM_RAIDEnumeration (see Section 16.1), DCIM_RAIDInteger (this section), and DCIM_RAIDString (see Section 16.5).

Enumerate the DCIM_RAIDInteger class to display all the RAID controller attributes in a system.

Enumerate RAIDInteger with the following parameters and syntax:

**EXAMPLE**:

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim
```
The 'get' instance method in Section 16.4 used this InstanceID as input.

The 'set attribute' method in Section 16.19.3 uses the FQDD, AttributeName, and a value equal to or between the LowerBound and UpperBound fields as input.

The 'set attributes' method in section 16.19.4 uses the FQDD, AttributeName, and a value equal to or between the LowerBound and UpperBound fields as input.
16.4 Getting a RAID Integer Instance

Use the following example to get an instance of the DCIM_RAIDInteger class, instead of all instances as shown in Section 16.3.

Get a RAIDInteger instance with the following parameters and syntax:

```
[INSTANCEID]: This is obtained from the enumeration in Section 16.3, which shows an example using RAID.Integrated.1-1:RAIDrebuildRate as an instanceID
```

**EXAMPLE:**

```
winrm g cimv2/root/dcim/DCIM_RAIDInteger?instancesID=[INSTANCE ID]
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman
-SkipCNcheck
-SkipCAcheck
-encoding:utf-8
-a:basic
```

**OUTPUT:**

```
DCIM_RAIDInteger
  AttributeName = RAIDrebuildRate
  CurrentValue = 30
  FQDD = RAID.Integrated.1-1
  InstanceID = RAID.Integrated.1-1:RAIDrebuildRate
  IsReadOnly = false
  LowerBound = 1
  PendingValue
  UpperBound = 100
```

16.5 Listing the RAID Inventory-String Class

The RAID Inventory has these attributes: DCIM_RAIDEnumeration (see Section 16.1), DCIM_RAIDInteger (see Section 16.3), and DCIM_RAIDString (this section).

Enumerate the DCIM_RAIDString class to display all the RAID controller string attributes in a system.

Enumerate RAIDString with the following parameters and syntax:

**EXAMPLE:**

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim
-schema/2/root/dcim/DCIM_RAIDString
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443
-SkipCNcheck
-SkipCACheck
-encoding:utf-8
-a:basic
```
OUTPUT:

DCIM_RAIDString
  AttributeName = Name
  CurrentValue = MyCacheCadeVD
  FQDD = Disk.Virtual.0:RAID.Integrated.1-1
  InstanceID = Disk.Virtual.0:RAID.Integrated.1-1:Name
  IsReadOnly = true
  MaxLength = 15
  MinLength = 0
  PendingValue

DCIM_RAIDString
  AttributeName = Name
  CurrentValue = raid 1 vd
  FQDD = Disk.Virtual.1:RAID.Integrated.1-1
  InstanceID = Disk.Virtual.1:RAID.Integrated.1-1:Name
  IsReadOnly = true
  MaxLength = 15
  MinLength = 0
  PendingValue

16.6 Getting a RAID String Instance

Use the following example to get an instance of the DCIM_RAIDString class instead of all instances as shown in Section 16.5.

Get a DCIM_RAIDString instance with the following parameters and syntax:

```
winrm g cimv2/root/dcim/DCIM_RAIDString?InstanceID=[INSTANCE ID]
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAncheck
-encoding:utf-8 -a:basic
```

EXAMPLE:

```
winrm g cimv2/root/dcim/DCIM_RAIDString?InstanceID=[INSTANCE ID]
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAncheck
-encoding:utf-8 -a:basic
```

OUTPUT:

DCIM_RAIDString
  AttributeName = Name
  CurrentValue = MyCacheCadeVD
  FQDD = Disk.Virtual.0:RAID.Integrated.1-1
  InstanceID = Disk.Virtual.0:RAID.Integrated.1-1:Name
  IsReadOnly = true
  MaxLength = 15
  MinLength = 0

The ‘get’ instance method in Section 16.6 uses this InstanceID as input.
16.7 Listing the RAID Inventory-ControllerView Class

The `DCIM_ControllerView` class groups together a set of Controller properties.

Enumerate `ControllerView` with the following parameters and syntax:

**EXAMPLE:**

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_ControllerView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

```
DCIM_ControllerView
  Bus = 1
  CacheSizeInMB = 0
  CacheCascadeCapability = 0
  ControllerFirmwareVersion = 20.10.1-0066
  Device = 0
  DeviceCardDataBusWidth = 1
  DeviceCardManufacturer = DELL
  DeviceCardSlotLength = 4
  DeviceCardSlotType = PCI Express x8
  DriverVersion = null
  EncryptionCapability = 0
  EncryptionMode = 0
  FQDD = RAID.Slot.1-1
  Function = 0
  InstanceID = RAID.Slot.1-1
  KeyID = null
  LastSystemInventoryTime = 20120116145459.000000+000
  LastUpdateTime = 20120116145459.000000+000
  PCIDeviceID = 73
  PCISlot = 1
  PCISubDeviceID = 1F4E
  PCISubVendorID = 1028
  PCIVendorID = 1000
  PatrolReadState = 1
  PrimaryStatus = 1
  ProductName = PERC H310 Adapter
  RollupStatus = 1
  SASAddress = 5782BCB00C577600
  SecurityStatus = 0
  SlicedVDCapability = 1
```

The ‘get’ instance method in Section Error! Reference source not found. will use this `InstanceID` as input.
16.8 Getting a RAID ControllerView Instance

The get() command can be invoked using a particular instanceId, attained from listing the inventory.

Get a RAID ControllerView instance with the following parameters and syntax:

```
[INSTANCEID]: This is obtained from the enumeration in Section 16.7, in which this example would use RAID.Slot.1-1 as an instanceId
```

**EXAMPLE:**

```
winrm g http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_ControllerView
?InstanceID=[[INSTANCEID]]
-r:https://[IPADDRESS]:443/wsman
-u:[USER] -p:[PASSWORD]
-auth:basic -encoding:utf-8 -SkipCNCheck -SkipCACheck
```

**OUTPUT:**

```
DCIM_ControllerView
  Bus = 1
  CacheSizeInMB = 0
  CachecadeCapability = 0
  ControllerFirmwareVersion = 20.10.1-0066
  Device = 0
  DeviceCardDataBusWidth = 1
  DeviceCardManufacturer = DELL
  DeviceCardSlotLength = 4
  DeviceCardSlotType = PCI Express x8
  DriverVersion = null
  EncryptionCapability = 0
  EncryptionMode = 0
  FQDD = RAID.Slot.1-1
  Function = 0
  InstanceID = RAID.Slot.1-1
  KeyID = null
  LastSystemInventoryTime = 20120116145459.000000+000
  LastUpdateTime = 20120116145459.000000+000
  PCIDeviceID = 73
  PCISlot = 1
  PCISubDeviceID = 1F4E
  PCISubVendorID = 1028
  PCIVendorID = 1000
  PatrolReadState = 1
  PrimaryStatus = 1
  ProductName = PERC H310 Adapter
  RollupStatus = 1
  SASAddress = 5782BCB00C577600
  SecurityStatus = 0
  SlicedVDCapability = 1
```
16.9 Listing the RAID Inventory-PhysicalDiskView Class

Enumerating the PhysicalDiskView, results in the attributes and inventory of the available physical disks in the system.

Enumerate PhysicalDiskView with the following parameters and syntax:

**EXAMPLE:**

```
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

```
DCIM_PhysicalDiskView
BusProtocol = 6
Connector = 0
DriveFormFactor = 3
FQDD = Disk.Bay.0:Enclosure.Internal.0-0:RAID.Slot.1-1
FreeSizeInBytes = 8978432
HotSpareStatus = 0
InstanceID = Disk.Bay.0:Enclosure.Internal.0-0:RAID.Slot.1-1
LastSystemInventoryTime = 20120116145459.000000+000
LastUpdateTime = 20120116145459.000000+000
Manufacturer = SEAGATE
ManufacturingDay = 7
ManufacturingWeek = 50
ManufacturingYear = 2010
MaxCapableSpeed = 3
MediaType = 0
Model = ST9500430SS
OperationName = None
OperationPercentComplete = 0
PPID = TH0R734K212330CG0027A00
PredictiveFailureState = 0
PrimaryStatus = 1
RaidStatus = 2
Revision = DS62
RollupStatus = 1
SASAddress = 5000C50025D64875
SecurityState = 0
SerialNumber = 9SP297S1
SizeInBytes = 499558383616
Slot = 0
SupportedEncryptionTypes = None
UsedSizeInBytes = 35827154944

DCIM_PhysicalDiskView
BusProtocol = 6
Connector = 0
DriveFormFactor = 2
```

152
**16.10 Listing the RAID VirtualDiskView Inventory**

Enumerating the VirtualDiskView, results in the attributes and inventory of the available virtual disks in the system.

Enumerate VirtualDiskView with the following parameters and syntax:

```
EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_VirtualDiskView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

```
DCIM_VirtualDiskView
   BusProtocol = 6
   Cachecade = 0
   DiskCachePolicy = 1024
   FQDD = Disk.Virtual.0:RAID.Slot.1-1
   InstanceID = Disk.Virtual.0:RAID.Slot.1-1
```
Virtual disks will denote 3 (pending) prior to being created, and 0 after creation
StripeSize = 128
VirtualDiskTargetID = 0
WriteCachePolicy = 1

16.11 Listing the RAID EnclosureView Inventory

Enumerating the EnclosureView, results in the attributes and inventory of the available enclosure components in the system.

Enumerate EnclosureView with the following parameters and syntax:

**EXAMPLE:**

```bash
```

**OUTPUT:**

```
DCIM_EnclosureView
   AssetTag
   Connector = 0
   EMMCount = 0
   FQDD = Enclosure.Internal.0-0:RAID.Integrated.1-1
   FanCount = 0
   InstanceID = Enclosure.Internal.0-0:RAID.Integrated.1-1
   LastSystemInventoryTime = 20100413194610
   LastUpdateTime = 20100413193143
   PSUCount = 0
   PrimaryStatus = 0
   ProductName = BACKPLANE 0:0
   RollupStatus = 0
   ServiceTag
   SlotCount = 6
   TempProbeCount = 0
   Version = 1.07
   WiredOrder = 0
```

16.12 Reset Configuration-ResetConfig()

The ResetConfig() method is used to delete all virtual disks and unassign all HotSpare physical disks. The deletions will not occur until a configuration job (Section 16.15) is scheduled and the system is rebooted. **All data on the existing virtual disks will be lost!**

Invoke ResetConfig with the following parameters and syntax:

**TARGET:** This parameter is the FQDD of the DCIM_ControllerView (Section 16.7)

**EXAMPLE:**
The input file `ResetConfig.xml` is shown below:

```xml
  <p:Target>RAID.Integrated.1-1</p:Target>
</p:ResetConfig_INPUT>
```

**OUTPUT:**

ResetConfig_OUTPUT

    ReturnValue = 0

### 16.13 Clearing the Foreign Configuration-ClearForeignConfig()

The `ClearForeignConfig()` method is used to prepare any foreign physical disks for inclusion in the local configuration.

Invoke `ClearForeignConfig()` with the following parameters and syntax:

**TARGET:** This parameter is the FQDD of the `DCIM_ControllerView` ([Section 16.7](#)).

**EXAMPLE:**

```
winrm i ClearForeignConfig cimv2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_RAIDService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCACheck
-encoding:utf-8 -a:basic -file:ClearForeignConfig.xml
```

The input file `ClearForeignConfig.xml` is shown below:

```xml
  <p:Target>RAID.Integrated.1-1</p:Target>
</p:ClearForeignConfig_INPUT>
```

**OUTPUT:**

ClearForeignConfig_OUTPUT

    ReturnValue = 0
If no foreign physical disks are available, the following message may result:

ClearForeignConfig_OUTPUT
   Message = General failure
   MessageID = STOR006
   ReturnValue = 2

16.14 Applying the Pending Values for RAID

The CreateTargetedConfigJob() method is called to apply the pending values created by RAID methods. The system will automatically reboot depending on the ScheduledStartTime selected. The CreateTargetedConfigJob() jobID output with the job control section can be used to obtain its status.

Invoke CreateTargetedConfigJob() with the following parameters and syntax:

   TARGET:  This parameter is the FQDD of the DCIM_ControllerView (Section 16.7)

   RebootJobType:  There are three options for rebooting the system.

      1 = PowerCycle
      2 = Graceful Reboot without forced shutdown
      3 = Graceful reboot with forced shutdown

   Note:  When a user does not want to set a reboot type when creating a target job, users should comment out the RebootJobType in the input xml. User should not enter “0” or give no parameter at all in the input xml.

   ScheduledStartTime & UntilTime:  See Section 3.2.4

EXAMPLE:

   winrm i CreateTargetedConfigJob http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_RAIDService
   ?SystemCreationClassName=DCIM_ComputerSystem
   +CreationClassName=DCIM_RAIDService
   +SystemName=DCIM:ComputerSystem
   +Name=DCIM:RAIDService
   -u:[USER] -p:[PASSWORD]
   -r:http://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
   -encoding:utf-8 -a:basic -file:CreateTargetedConfigJob_RAID.xml

The input file CreateTargetedConfigJob_RAID.xml is shown below:

  <p:Target>RAID.Integrated.1-1</p:Target>
  <p:RebootJobType>3</p:RebootJobType>
  <p:ScheduledStartTime>TIME_NOW</p:ScheduledStartTime>
  <p:UntilTime>2011111111111</p:UntilTime>
</p:CreateTargetedConfigJob_INPUT>

**OUTPUT:**

When this method is executed, a *jobid* or an error message is returned. The status of this *jobid* can be checked within the job control provider in **Section 10**.

CreateTargetedConfigJob_OUTPUT

```
Job
   Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
   ReferenceParameters
      SelectorSet
         Selector: InstanceID = JID_001271251761, __cimnamespace = root/dcim
   ReturnValue = 4096
```

**16.15 Deleting the Pending Values for RAID-DeletePendingConfiguration()**

The *DeletePendingConfiguration()* method cancels the pending configuration changes made before the configuration job is created with *CreateTargetedConfigJob()*. This method only operates on the pending changes prior to *CreateTargetedConfigJob()* being called. After the configuration job is created, the pending changes can only be canceled by calling *DeleteJobQueue()* in the Job Control profile.

Invoke *DeletePendingConfiguration()* with the following parameters and syntax:

```
TARGET: This parameter is the FQDD of the DCIM_ControllerView (Section 16.7)
```

**EXAMPLE:**

```
winrm i DeletePendingConfiguration http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_RAIDService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:RAIDService
-u: [USER] -p: [PASSWORD]
-r: https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a: basic -file:DeletePendingConfiguration.xml
```

The input file *DeletePendingConfiguration.xml* is shown below:

```
   <p:Target>RAID.Integrated.1-1</p:Target>
</p:DeletePendingConfiguration_INPUT>
```

**OUTPUT:**

DeletePendingConfiguration_OUTPUT

```
   ReturnValue = 0
```

158
16.16 Managing Hot Spare

16.16.1 Determining Potential Disks-GetDHSDisks()
The GetDHSDisks() method is used to determine possible choices of drives to be a dedicated HotSpare for the identified virtual disk.

Invoke GetDHSDisks() with the following parameters and syntax:

TARGET: This parameter is the FQDD of the target virtual disk. Its value will depend on the number of virtual disks, obtainable in Section 16.10.

EXAMPLE:

winrm i GetDHSDisks cimv2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
+CreationClassName=DCIM_RAIDService
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:GetDHSDisks.xml

The input file GetDHSDisks.xml is shown below:

```xml
    <p:Target>DISK.Virtual.1:RAID.Integrated.1-1</p:Target>
</p:GetDHSDisks_INPUT>
```

OUTPUT:

GetDHSDisks_OUTPUT
    ReturnValue = 0

The following message may be fixed by deleting the job queue as referenced in Section 10.2.2.

GetDHSDisks_OUTPUT
    Message = Configuration already committed, cannot set configuration
    MessageID = STOR023
    ReturnValue = 2

16.16.2 Assigning the Hot Spare.AssignSpare()
The AssignSpare() method is used to assign a physical disk as a dedicated HotSpare for a virtual disk (VD), or as a global HotSpare.

Invoke AssignSpare() with the following parameters and syntax:

TARGET: This parameter is the FQDD of the DCIM_PhysicalDiskView (Section 16.9)
**VirtualDiskArray:** Array of ElementName(s) where each identifies a different VD, currently only one VD can be passed

**EXAMPLE:**

```plaintext
winrm i AssignSpare http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_RAIDService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAscheck
-encoding:utf-8 -a:basic -file:AssignSpare.xml
```

The input file **AssignSpare.xml** is shown below:

```xml
    <p:VirtualDiskArray>Disk.Virtual.0 :RAID.Integrated.1-1</p:VirtualDiskArray>
</p:AssignSpare_INPUT>
```

**OUTPUT:**

AssignSpare_OUTPUT
  RebootRequired = YES
  ReturnValue = 0

Nonconformance to the following restrictions may result in the error message below.

- Virtual disk (VD) referenced (dedicated hot spare) is RAID-0, which cannot have hot spares
- Physical disk (PD) is too small for the virtual disk referenced (dedicated hot spare)
- Physical disk is wrong type for the virtual disk (i.e. SATA PD to be used as hot spare for SAS VD)
- Similar conditions when no VD referenced, which is the global hot spare attempted assignment

**ERROR MESSAGE:**

AssignSpare_OUTPUT
  Message = Physical disk FQDD did not identify a valid physical disk for the operation
  MessageID = STOR009
  ReturnValue = 2
16.16.3 Unassigning the Hot Spare - UnassignSpare()

The UnassignSpare() method is used to unassign a physical disk. The physical disk may be used as a dedicated hot spare to a virtual disk, or as a global hot spare. After the method executes successfully the physical disk is no longer a hotspare.

Invoke UnassignSpare() with the following parameters and syntax:

   TARGET: This parameter is the FQDD of the DCIM_PhysicalDiskView (Error! Reference source not found.)

EXAMPLE:

winrm i UnassignSpare
cimv2/root/dcim/DCIM_RAIDService
    +SystemCreationClassName=DCIM_ComputerSystem
    +CreationClassName=DCIM_RAIDService
    +SystemName=DCIM:ComputerSystem+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
    -encoding:utf-8 -a:basic -file:UnassignSpare.xml

The input file UnassignSpare.xml is shown below:

<p:UnassignSpare_INPUT
  xmlns:p="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_RAIDService">
</p:UnassignSpare_INPUT>

OUTPUT:

UnassignSpare_OUTPUT
  RebootRequired = YES
  ReturnValue = 0

16.17 Managing Keys for Self Encrypting Drives

NOTE: The Dell Key Manager feature is not available at this time.

16.17.1 Setting the Key - SetControllerKey()

The SetControllerKey() method sets the key on controllers that support encryption of the virtual disk drives.

Invoke SetControllerKey() with the following parameters and syntax:

   TARGET: This parameter is the FQDD of the DCIM_ControllerView (Section 16.7)
   Key: Maximum size 32 characters
   Keyid: Identifier, or description, for the key (maximum size 255 characters)
**EXAMPLE:**

```
winrm i SetControllerKey http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_RAIDService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:SetControllerKey.xml
```

The input file `SetControllerKey.xml` is shown below:

```xml
  <p:Target>RAID.Integrated.1-1</p:Target>
  <p:Key>abc123</p:Key>
  <p:KeyId>keyid</p:KeyId>
</p:SetControllerKey_INPUT>
```

**OUTPUT:**

This method requires an H700 or H800 controller to properly function. Running this method on older controllers may yield this message:

```
SetControllerKey_OUTPUT
  Message = Controller is not security capable
  MessageID = STOR022
  ReturnValue = 2
```

**16.17.2 Locking the Virtual Disk-LockVirtualDisk()**

The `LockVirtualDisk()` method encrypts the virtual disk. Note that the virtual disk must first exist for this method to be successful.

Invoke `LockVirtualDisk()` with the following parameters and syntax:

```
TARGET: This parameter is the FQDD of the target virtual disk
```

**EXAMPLE:**

```
winrm i LockVirtualDisk http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_RAIDService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:LockVirtualDisk.xml
```
The input file LockVirtualDisk.xml is shown below:

```xml
<p:Target>Disk.Virtual.0:RAID.Integrated.1-1</p:Target>
</p:LockVirtualDisk_INPUT>
```

**OUTPUT:**

This method requires an H700 or H800 controller to properly function, as does the LockVirtualDisk() method. If the key is not set by LockVirtualDisk(), the following message may be displayed:

LockVirtualDisk_OUTPUT
Message = Controller Key is not present
MessageID = STOR021
ReturnValue = 2

16.17.3 Locking the Controller with a Key-EnableControllerEncryption()

The EnableControllerEncryption() method is used to set either Local Key encryption or Dell Key Manager (DKM) encryption on controllers that support encryption of the drives.

Invoke EnableControllerEncryption() method with the following parameters and syntax:

**TARGET:** This parameter is the FQDD of the DCIM_ControllerView class. See Section 16.1.

**Key:** Key – Passcode. This parameter is required if the Mode = Local Key Encryption. The Key can be maximum 32 characters in length, and must have one character from each of the following sets.
- Upper Case
- Lower Case
- Number
- Special Character

The special characters in the following set needs to be passed as mentioned below.
& → &amp;
< → &lt;
> → &gt;
“ → &quot;
' → &apos;

**Keyid:** Key Identifier - Describes Key. The Keyid can be maximum 32 characters in length and must not have spaces in it.

**Mode:** Mode of the Controller
1 - Local Key Encryption
2 – Dell Key Manager
EXAMPLE:

winrm i EnableControllerEncryption
http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/root/dcim/DCIM_RAIDService?SystemCreationCl
assName=DCIM_ComputerSystem
+CreationClassName=DCIM_RAIDService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
-file:EnableControllerEncryption.xml

The information in the input file EnableControllerEncryption.xml is shown below:

<p:EnableControllerEncryption_INPUT xmlns:p="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/root/dcim/DCIM_RAIDService">  
<p:Target>RAID.Integrated.1-1</p:Target>  
<p:Mode>1</p:Mode>  
<p:Key>Abcd@123</p:Key>  
<p:Keyid>LKM</p:Keyid>  
</p:EnableControllerEncryption_INPUT>

OUTPUT:

This method requires an PERC controller with Local Key encryption or DKM support to function correctly.

EnableControllerEncryption_OUTPUT  
  RebootRequired = YES  
  ReturnValue = 0

16.17.4 Rekeying the Controller-ReKey()  
The ReKey() method is used to reset the key on the controller that supports encryption. This method switches the controller mode between Local Key encryption or Dell Key Manager (DKM) encryption.

Invoke the ReKey() method with the following parameters and syntax:

TARGET: This parameter is the FQDD of the DCIM_ControllerView class. See section 16.1.

OldKey: Old controller key

NewKey: New controller key. The Key can be maximum 32 characters long, and must have one character from each of the following:
  Upper Case
  Lower Case
  Number
  Special Character
The special characters in the following set must be passed as mentioned below.
& → &amp;
< → &lt;
> → &gt;
“ → &quot;
‘ → &apos;

**Keyid:** Key Identifier - Describes Key. The Keyid can be maximum 32 characters long and should not have spaces in it.

**Mode:** Mode of the Controller
1 - Local Key Encryption
2 – Dell Key Manager

**EXAMPLE:**

```
winrm i ReKey
cimv2/root/dcim/DCIM_RAIDService?SystemCreationClassName=DCIM_ComputerSystem+CreationClassName=DCIM_RAIDService+SystemName=DCIM:ComputerSystem+Name=DCIM:RAIDService
-u:[USER]
-p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:ReKey.xml
```

The information in the input file *ReKey.xml* is shown below:

```
<p:ReKey_INPUT xmlns:p="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_RAIDService">
  <p:Target>RAID.Integrated.1-1</p:Target>
  <p:OldKey>Abcd@123</p:OldKey>
  <p:NewKey>Efgh@123</p:NewKey>
  <p:Keyid>NewLKMid</p:Keyid>
  <p:Mode>1</p:Mode>
</p:ReKey_INPUT>
```

**OUTPUT:**

This method requires a PERC controller with Local Key encryption or DKM support to function correctly.
If the `EnableControllerEncryption()` method does not set the key, the following message is displayed:

```
ReKey_OUTPUT
  Message = Controller Key is not present
  MessageID = STOR021
  ReturnValue = 2
```
16.17.5 Removing the Key-RemoveControllerKey()
The RemoveControllerKey() method is used to erase the key on the controller along with the attached encrypted drives.

Invoke the RemoveControllerKey() method with the following parameters and syntax:

**TARGET**: This parameter is the FQDD of the DCIM_ControllerView class. See section 16.1.

**EXAMPLE**:
```bash
winrm i RemoveControllerKey

cimv2/root/dcim/DCIM_RAIDService?SystemCreationClassName=DCIM_ComputerSystem+CreationClassName=DCIM_RAIDService+SystemName=DCIM:ComputerSystem+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -file:RemoveControllerKey.xml
```

The input file RemoveControllerKey.xml is shown below:
```xml
  <p:Target>
    RAID.Integrated.1
  </p:Target>
</p:RemoveControllerKey_INPUT>
```

**OUTPUT**:
This method requires an H700 or H800 controller to function correctly. If the EnableControllerEncryption() method does not set the key, the following message is displayed:

RemoveControllerKey_OUTPUT
Message = Controller Key is not present
MessageID = STOR021
ReturnValue = 2

16.18 Managing Virtual Disk

16.18.1 Getting the Available RAID levels-GetRAIDLevels()
The GetRAIDLevels() method is used to determine possible choices RAID levels to create virtual disks. If the list of physical disks is not provided, this method will operate on all connected disks.

Invoke GetRAIDLevels() with the following parameters and syntax:

**TARGET**: This parameter is the FQDD of the DCIM_ControllerView ([Section 16.7])

**DiskType**: Corresponds to MediaType attribute in PhysicalDiskView ([Section 16.9])

  Include all types=0, Include Magnetic Only=1, Include SSD only=2

**Diskprotocol**: Types of protocol to include
Include all protocols=0, Include SATA=1, Include SAStypes=2

**DiskEncrypt**: Types of encryption to include

- 0 = Include FDE capable and non encryption capable disks
- 1 = Include FDE disks only
- 2 = Include only non FDE disks

**PDArray**: This parameter is the list of physical disk FQDDs

**EXAMPLE**:

```
winrm i GetRAIDLevels cimv2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
+CreationClassName=DCIM_RAIDService
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:GetRAIDLevels.xml
```

The input file **GetRAIDLevels.xml** is shown below:

```
  <p:Target>RAID.Integrated.1-1</p:Target>
  <p:DiskType>0</p:DiskType>
  <p:Diskprotocol>0</p:Diskprotocol>
  <p:DiskEncrypt>0</p:DiskEncrypt>
  <p:PDArray>Disk.Bay.0:Enclosure.Internal.0:RAID.Integrated.1-1</p:PDArray>
</p:GetRAIDLevels_INPUT>
```

**OUTPUT**:

GetRAIDLevels_OUTPUT

Return Value = 0
VDRAIDEnumArray = 2, 4

The **VDRAIDEnumArray** numbers correspond to the following RAID levels:

```
RAIDLevel:

RAID 0 = 2
RAID 1 = 4
RAID 5 = 64
RAID 6 = 128
```
RAID 10 = 2048
RAID 50 = 8192
RAID 60 = 16384

16.18.2 Getting the Available Disks-GetAvailableDisks()
The GetAvailableDisks() method is used to determine possible choices of drives to create virtual disks.

Invoke GetAvailableDisks() with the following parameters and syntax:

TARGET: This parameter is the FQDD of the DCIM_ControllerView (Section 16.7)

DiskType: Corresponds to MediaType attribute in PhysicalDiskView (Section 16.9)
Include all types=0, Include Magnetic Only=1, Include SSD only=2

Diskprotocol: Types of protocol to include
Include all protocols=0, Include SATA=1, Include SAS types=2

DiskEncrypt: Types of encryption to include
0 = Include FDE capable and non encryption capable disks
1 = Include FDE disks only
2 = Include only non FDE disks

EXAMPLE:

winrm i GetAvailableDisks cimv2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
  +SystemName=DCIM:ComputerSystem
  +CreationClassName=DCIM_RAIDService
  +Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:GetAvailableDisks.xml

The input file GetAvailableDisks.xml is shown below:

```xml
  <p:Target>RAID.Integrated.1-1</p:Target>
  <p:DiskType>0</p:DiskType>
  <p:Diskprotocol>0</p:Diskprotocol>
  <p:DiskEncrypt>0</p:DiskEncrypt>
  <p:Raidlevel>2</p:Raidlevel>
</p:GetAvailableDisks_INPUT>
```
OUTPUT:

GetAvailableDisks_OUTPUT
  ReturnValue = 0

16.18.3 Checking the Create VD Parameters Validity - CheckVDValues()

The CheckVDValues() method is used to determine possible sizes of virtual disk as well default settings, given a RAID level and set of disks. The VDPropArray is filled in with Size and other values for a successful execution of the method.

Invoke CheckVDValues() with the following parameters and syntax:

TARGET:  This parameter is the FQDD of the DCIM_ControllerView (Section 16.7)

PDArray:  This parameter is the list of physical disk FQDDs (Section 16.9)

VDPropNameArrayIn:  This parameter is the list of property names with values in the VDPropValueArrayIn parameter

Size, RAIDLevel, SpanDepth

VDPropValueArrayIn:  This parameter is the list of property values that correspond to the VDPropNameArrayIn parameter

EXAMPLE:

winrm i CheckVDValues cimv2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+SystemName=DCIM:ComputerSystem
+CreationClassName=DCIM_RAIDService
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:CheckVDValues.xml

The input file CheckVDValues.xml is shown below:

<p:CheckVDValues_INPUT xmlns:p="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_RAIDService">
  <p:Target>RAID.Integrated.1-1</p:Target>
</p:CheckVDValues_INPUT>
16.18.4 Creating a Single Virtual Disk—CreateVirtualDisk()

The CreateVirtualDisk() method is used to create a single virtual disk on the targeted controller. The successful execution of this method results in a pending but not yet created virtual disk. The ObjectStatus property in the virtual disk view (Section 16.10) will have the value ‘3’, which represents pending. The virtual disk will not be created until a configuration job (Section 16.15) has been scheduled and the system is rebooted. Upon creation of the virtual disk, the FQDD of the formerly pending virtual disk will change.

Invoke CreateVirtualDisk() with the following parameters and syntax:

**TARGET:** This parameter is the FQDD of the DCIM_ControllerView (Section 16.7)

**PDArray:** This parameter is the list of physical disk FQDDs that will be used to create a virtual Disk.

**VDPropNameArray:** This parameter is the list of property names that will be used to create a virtual disk. The parameter list contains the following names:

- Size, RAIDLevel, SpanDepth, SpanLength, StripeSize, ReadPolicy, WritePolicy, DiskCachePolicy, VirtualDiskName, Initialize

**VDPropValueArray:** This parameter is the list of property values that will be used to create a virtual Disk. The property values are for the property names listed under VDPropNameArray.
Size: Size of the virtual disk specified in MB. If not specified, default will use full size of physical disks selected.

RAIDLevel:

RAID 0 = 2
RAID 1 = 4
RAID 5 = 64
RAID 6 = 128
RAID 10 = 2048
RAID 50 = 8192
RAID 60 = 16384

SpanDepth: If not specified, default is single span which is used for RAID 0, 1, 5 and 6. Raid 10, 50 and 60 require a spandepth of at least 2.

SpanLength: Number of Physical Disks to be used per span. Minimum requirements for given RAID Level must be met.

StripeSize:

8KB = 16
16KB = 32
32KB = 64
64KB = 128
128KB = 256
256KB = 512
512KB = 1024
1MB = 2048
ReadPolicy:

- No Read Ahead = 16
- Read Ahead = 32
- Adaptive Read Ahead = 64

WritePolicy:

- Write Through = 1
- Write Back = 2
- Write Back Force = 4

DiskCachePolicy:

- Enabled = 512
- Disabled = 1024

VirtualDiskName: Name of the virtual disk (1-15 character range)

Example:

```
winrm i CreateVirtualDisk cimv2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_RAIDService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:CreateVirtualDisk.xml
```

The input file `CreateVirtualDisk.xml` is shown below:

```
  <p:Target>RAID.Integrated.1-1</p:Target>
  <p:VDPropNameArray>RAIDLevel</p:VDPropNameArray>
  <p:VDPropNameArray>SpanDepth</p:VDPropNameArray>
</p:CreateVirtualDisk_INPUT>
```
<p:VDPropNameArray>SpanLength</p:VDPropNameArray>
<p:VDPropNameArray>Size</p:VDPropNameArray>
<p:VDPropNameArray>VirtualDiskName</p:VDPropNameArray>
<p:VDPropValueArray>4</p:VDPropValueArray>
<p:VDPropValueArray>1</p:VDPropValueArray>
<p:VDPropValueArray>2</p:VDPropValueArray>
<p:VDPropValueArray>100</p:VDPropValueArray>
<p:VDPropValueArray>virtualdiskname</p:VDPropValueArray>
</p:CreateVirtualDisk_INPUT>

**OUTPUT:**

The `instanceID` output will identify this virtual disk in inventory before and after its creation by the `CreateTargetedConfigJob`. Note however, that the `instanceID` will change slightly after successful creation.

**CreateVirtualDisk_OUTPUT**

NewVirtualDisk
- ReferenceParameters
  - SelectorSet
    - Selector: `InstanceID = DISK.Virtual.267386880:RAID.Integrated.1-1`, `__cimnamespace = root/dcim`
- RebootRequired = YES
- ReturnValue = 0

### 16.18.5 Creating a Sliced Virtual Disk—CreateVirtualDisk()

The `CreateVirtualDisk()` method is used to create a sliced virtual disk. A sliced virtual disk is created, if `CreateVirtualDisk()` size input parameter value is less than total size of the physical disks. Additional sliced virtual disk can be created using the same set of physical disks and same RAID level that was used to create the first sliced virtual disk. If the physical disks have sliced virtual disks, then use the `CheckVDValues()` method on that set of physical disks to find the exact value for StartingLBA. Use this value as the `StartingLBA` parameter value of the `CreateVirtualDisk()` method.

The `ObjectStatus` property in the virtual disk view (see Section 16.10) has the value ‘3’, which represents a pending change. The virtual disk is not created until a configuration job (see Section 16.14) is scheduled and the system is rebooted. After the virtual disk creation, the FQDD of the pending virtual disk changes.

Invoke the `CreateVirtualDisk()` method with the following parameters and syntax:

**TARGET:** This parameter is the FQDD of the `DCIM_ControllerView` (Section 16.7)

**PDArray:** This parameter is the list of physical disk FQDDs that is used to create a virtual Disk.

**VDPropNameArray:** This parameter is the list of property names that is used to create a virtual disk. The parameter list has the following names:
Size, RAIDLevel, SpanDepth, SpanLength, StripeSize, ReadPolicy, WritePolicy, DiskCachePolicy, VirtualDiskName, Initialize

**VDPropValueArray**: This parameter is the list of property values that is used to create a virtual Disk. The property values are for the property names listed under *VDPropNameArray*.

- **Size**: Size of the virtual disk specified in MB. If not specified, default will use full size of physical disks selected.

- **RAIDLevel**:
  - RAID 0 = 2
  - RAID 1 = 4
  - RAID 5 = 64
  - RAID 6 = 128
  - RAID 10 = 2048
  - RAID 50 = 8192
  - RAID 60 = 16384

- **SpanDepth**: If not specified, default is single span which is used for RAID 0, 1, 5 and 6. Raid 10, 50 and 60 require a spandepth of at least 2.

- **SpanLength**: Number of Physical Disks to be used per span. Minimum requirements for given RAID Level must be met.

- **StripeSize**:
  - 8KB = 16
  - 16KB = 32
  - 32KB = 64
  - 64KB = 128
  - 128KB = 256
256KB = 512
512KB = 1024
1MB = 2048

ReadPolicy:

No Read Ahead = 16
Read Ahead = 32
Adaptive Read Ahead = 64

WritePolicy:

Write Through = 1
Write Back = 2
Write Back Force = 4

DiskCachePolicy:

Enabled = 512
Disabled = 1024

VirtualDiskName: Name of the virtual disk (1-15 character range)

StartingLBA: Starting logical block address of virtual disks in blocks. If 0xFFFFFFFFFFFFFFFF, startingLBA is calculated programatically. The value can be in hexadecimal or decimal format.

0xFFFFFFFFFFFFFFFF
18446744073709551615

EXAMPLE:

winrm i CreateVirtualDisk cimv2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_RAIDService
+VirtualDiskName=VirtualDiskName
+StartingLBA=StartingLBA
The input file `CreateSlicedVirtualDisk.xml` is shown below:

```xml
  <p:Target>RAID.Integrated.1-1</p:Target>
  <p:VDPropNameArray>RAIDLevel</p:VDPropNameArray>
  <p:VDPropNameArray>SpanDepth</p:VDPropNameArray>
  <p:VDPropNameArray>SpanLength</p:VDPropNameArray>
  <p:VDPropNameArray>Size</p:VDPropNameArray>
  <p:VDPropNameArray>VirtualDiskName</p:VDPropNameArray>
  <p:VDPropNameArray>StartingLBA</p:VDPropNameArray>
    <p:VDPropValueArray>4</p:VDPropValueArray>
    <p:VDPropValueArray>1</p:VDPropValueArray>
    <p:VDPropValueArray>2</p:VDPropValueArray>
    <p:VDPropValueArray>100</p:VDPropValueArray>
    <p:VDPropValueArray>virtualdiskname</p:VDPropValueArray>
    <p:VDPropValueArray>0xFFFFFFFFFFFFFFFF</p:VDPropValueArray>
</p:CreateVirtualDisk_INPUT>
```

**OUTPUT:**

The `instanceID` output identifies this virtual disk in the inventory before and after the `CreateTargetedConfigJob()` method creates it. However, the `instanceID` changes after successful creation.

CreateVirtualDisk_OUTPUT

NewVirtualDisk

```
Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
SelectorSet
  Selector: InstanceID = DISK.Virtual.267386880:RAID.Integrated.1-1, __cimnamespace = root/dcim
  RebootRequired = YES
  ReturnValue = 0
```

16.18.6 Creating a Cachecade Virtual Disk-CreateVirtualDisk()

The `CreateVirtualDisk()` method is used to create a Cachecade virtual disk on the targeted controller. This method internally creates a RAID-0 virtual disk. The creation process is the same as explained in Section 16.18.5. In this scenario, `CreateVirtualDisk()` method only takes `VDPropNameArray-VDPropValueArray` pairs mentioned below.
Invoke `CreateVirtualDisk()` with the following parameters and syntax:

**TARGET**: This parameter is the FQDD of the `DCIM_ControllerView` ([Section 16.7](#)).

**PDArray**: This parameter is the list of physical disk FQDDs that is used to create a virtual Disk.

**VDPropNameArray**: This parameter is the list of property names that is used to create a virtual disk. The parameter list has the following names:

- `VirtualDiskName`, `CacheCade`

**VDPropValueArray**: This parameter is the list of property values that is used to create a virtual Disk. The property values are for the property names listed under `VDPropNameArray`.

- `VirtualDiskName`: Name of the virtual disk (1-15 character range)
- `CacheCade`: The valid input value is 1. (required)

**EXAMPLE**:

```
winrm i CreateVirtualDisk cimv2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_RAIDService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:CreateVDCacheCade.xml
```

The input file `CreateVDCacheCade.xml` is shown below:

```
<p:CreateVirtualDisk_INPUT
 xmlns:p="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_RAIDService">
 <p:Target>RAID.Integrated.1-1</p:Target>
 <p:VDPropNameArray>VirtualDiskName</p:VDPropNameArray>
 <p:VDPropValueArray>MyCacheCadeVD</p:VDPropValueArray>
 <p:VDPropNameArray>Cachecade</p:VDPropNameArray>
 <p:VDPropValueArray>1</p:VDPropValueArray>
</p:CreateVirtualDisk_INPUT>
```

**OUTPUT**:

The `instanceID` output identifies this virtual disk in the inventory before and after the `CreateTargetedConfigJob()` method creates it. Note however, that the `instanceID` will change slightly after successful creation.

`CreateVirtualDisk_OUTPUT`
NewVirtualDisk
Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
SelectorSet
  Selector: InstanceID = DISK.Virtual.267386880:RAID.Integrated.1-1, __cimnamespace = root/dcim
  RebootRequired = YES
  ReturnValue = 0

16.18.7 Deleting a Virtual Disk-DeleteVirtualDisk()
The DeleteVirtualDisk() method is used to delete a single virtual disk from the targeted controller. The successful execution of this method results in the marking of this virtual disk for deletion. The ObjectStatus property in the virtual disk view will have the value of ‘2’, which indicates pending delete. The virtual disk will not be deleted until a configuration job is scheduled and the system is rebooted (Section 16.15).

Invoke DeleteVirtualDisk() with the following parameters and syntax:

  TARGET: This parameter is the FQDD of the virtual device (Section 16.10)

EXAMPLE:

winrm i DeleteVirtualDisk cimv2/root/dcim/DCIM_RAIDService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_RAIDService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding=utf-8 -a:basic -file:DeleteVirtualDisk.xml

The input file DeleteVirtualDisk.xml is shown below:

  <p:Target>DISK.Virtual.0:RAID.Integrated.1-1</p:Target>
</p:DeleteVirtualDisk_INPUT>

OUTPUT:

DeleteVirtualDisk_OUTPUT
  RebootRequired = YES
  ReturnValue = 0
16.19 Setting Controller Attributes

16.19.1 Changing the Value of a RAID Controller Enumeration Attribute
The SetAttribute() method is used to set or change the value of a RAID controller or a virtual disk attribute. The example below shows setting a RAID controller enumeration attribute. To set a virtual disk attribute, use the FQDD of the virtual disk attribute for the Target, and the AttributeName and AttributeValue.

Invoke SetAttribute() with the following parameters (from Section 16.1) and syntax:

```
TARGET: Obtained from the FQDD field
AttributeName: Obtained from the AttributeName field
AttributeValue: Obtained from the PossibleValues field
```

**EXAMPLE:**

```
```

The input file `SetAttribute_Enumeration_RAID_Controller.xml` is shown below:

```
  <p:Target>
    RAID.Integrated.1-1
  </p:Target>
  <p:AttributeName>RAIDBatteryLearnMode</p:AttributeName>
  <p:AttributeValue>Disabled</p:AttributeValue>
</p:SetAttribute_INPUT>
```

**OUTPUT:**

```
SetAttribute_OUTPUT
  Message = The method was successful.
  MessageID = STOR001
  RebootRequired = Yes
 ReturnValue = 0
SetResult = Set Pending Value
```

16.19.2 Changing Multiple Values of RAID Controller Enumeration Attributes
The SetAttributes() method is used to set or change multiple values of RAID controller or virtual disk attributes. The following example shows setting multiple virtual disk attributes. To set multiple controller attributes, use the FQDD of the controller for the Target, and the AttributeName and AttributeValue.

Invoke SetAttributes() with the following parameters (from Section 16.1) and syntax:
TARGET: Obtained from the FQDD field

AttributeName: Obtained from the AttributeName field

AttributeValue: Obtained from the PossibleValues field

EXAMPLE:

winrm i SetAttributes cimv2/root/dcim/DCIM_RAIDService?SystemCreationClassName=DCIM_ComputerSystem+CreationClassName=DCIM_RAIDService+SystemName=DCIM:ComputerSystem+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAncheck
-encoding:utf-8 -a:basic
-file:SetAttributesEnumeration_RAID_Controller.xml

The input file SetAttributesEnumeration_RAID_Controller.xml is shown below:

```xml
  <p:Target>RAID.Integrated.1-1</p:Target>
  <p:AttributeName>RAIDloadBalancedMode</p:AttributeName>
  <p:AttributeValue>Disabled</p:AttributeValue>
  <p:AttributeName>RAIDBatteryLearnMode</p:AttributeName>
  <p:AttributeValue>Warn only</p:AttributeValue>
  <p:AttributeName>RAIDccMode</p:AttributeName>
  <p:AttributeValue>Normal</p:AttributeValue>
  <p:AttributeName>RAIDprMode</p:AttributeName>
  <p:AttributeValue>Disabled</p:AttributeValue>
  <p:AttributeName>RAIDcopybackMode</p:AttributeName>
  <p:AttributeValue>SMART</p:AttributeValue>
</p:SetAttributes_INPUT>
```

OUTPUT:

```text
SetAttributes_OUTPUT
Message = The method was successful.
MessageID = STOR001
RebootRequired = Yes
ReturnValue = 0
SetResult = Set Pending Value
```

16.19.3 Changing the Value of a RAID Controller Integer Attribute

The SetAttribute() method is used to set or change the value of a RAID controller integer attribute. The example below shows setting an controller attribute.

Invoke the SetAttribute() method with the following parameters (from Section 16.1) and syntax:

TARGET: Obtained from the FQDD field

AttributeName: Obtained from the AttributeName field

AttributeValue: Obtained from the PossibleValues field
**EXAMPLE:**

```
winrm i SetAttribute cimv2/root/dcim/DCIM_RAIDService?
```

```
rvice?SystemCreationClassName=DCIM_ComputerSystem+CreationClassName=DCIM_RAIDService+System
Name=DCIM:ComputerSystem+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
-file:SetAttribute_Integer_RAID_Controller.xml
```

The input file `SetAttribute_Integer_RAID_Controller.xml` is shown below:

```xml
  <p:Target>
    RAID.Integrated.1-1
  </p:Target>
  <p:AttributeName>RAIDccRate</p:AttributeName>
  <p:AttributeValue>60</p:AttributeValue>
</p:SetAttribute_INPUT>
```

**OUTPUT:**

```
SetAttribute_OUTPUT
Message = The method was successful.
MessageID = STOR001
RebootRequired = Yes
ReturnValue = 0
SetResult = Set Pending Value
```

**16.19.4 Changing Multiple Values of RAID Controller Integer Attributes**

The `SetAttributes()` method is used to set or change multiple values of RAID controller attributes. The following example shows setting multiple RAID controller integer attributes.

Invoke `SetAttributes` with the following parameters (from Section 16.1) and syntax:

- **TARGET**: Obtained from the `FQDD` field
- **AttributeName**: Obtained from the `AttributeName` field
- **AttributeValue**: Obtained from the `PossibleValues` field

**EXAMPLE:**

```
winrm i SetAttributes cimv2/root/dcim/DCIM_RAIDService?
```

```
rvice?SystemCreationClassName=DCIM_ComputerSystem+CreationClassName=DCIM_RAIDService+System
Name=DCIM:ComputerSystem+Name=DCIM:RAIDService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
-file:SetAttributes=Integer_RAID_Controller.xml
```

The input file `SetAttributes_Integer_RAID_Controller.xml` is shown below:
  <p:Target>RAID.Integrated.1-1</p:Target>
  <p:AttributeName>RAIDccRate</p:AttributeName>
  <p:AttributeValue>60</p:AttributeValue>
  <p:AttributeName>RAIDreconstructRate</p:AttributeName>
  <p:AttributeValue>60</p:AttributeValue>
  <p:AttributeName>RAIDbgiRate</p:AttributeName>
  <p:AttributeValue>60</p:AttributeValue>
</p:SetAttributes_INPUT>

OUTPUT:

SetAttributes_OUTPUT
  Message = The method was successful.
  MessageID = STOR001
  RebootRequired = Yes
  ReturnValue = 0
  SetResult = Set Pending Value

### 16.20 Convert Physical Disks to RAID-ConvertToRAID()

The ConvertToRAID() method is used to convert physical disks in Non-RAID state to a state usable for RAID. After the method is successfully executed the PendingValue property of RAIDPDState should reflect the pending changes. After the CreateTargetedConfigJob() method is successfully executed the RAIDStatus property, which is enumerated in the DCIM_PhysicalDiskView from Section 16.9, of that physical disk should reflect the new state.

Invoke ConvertToRAID() with the following parameters and syntax:

**Physical Disk TARGET:** Obtained from the FQDD field (Section 16.9)

An example of Disk.Bay.2:Enclosure.Internal.0-0:RAID.Slot.1-1 is shown below.

**EXAMPLE:**

```
winrm invoke ConvertToRAID
"cimv2/root/dcim/DCIM_RAIDService?SystemCreationClassName=DCIM_ComputerSystem+CreationClassName=DCIM_RAIDService+SystemName=DCIM:ComputerSystem+Name=DCIM:RAIDService"
@{PDArray="Disk.Bay.2:Enclosure.Internal.0-0:RAID.Slot.1-1"}
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman
-SkipCNcheck -SkipCACheck -encoding:utf-8 -a:basic -format:pretty
```

OUTPUT:

ConvertToRAID_OUTPUT
  RebootRequired = 1
  ReturnValue = 0
16.21 Convert Physical Disks to Non RAID-ConvertToNonRAID()

The ConvertToNonRAID() method is used to convert a physical disks in RAID state of “Ready” to a Non-RAID state. After the method is successfully executed, the PendingValue property of RAIDPDState should reflect the pending changes. After the CreateTargetedConfigJob method is successfully executed, the RAIDStatus property, which is enumerated in the DCIM_PhysicalDiskView from Section 16.9, of that physical disk should reflect the new state.

Invoke ConvertToNonRAID() with the following parameters and syntax:

```plaintext
Physical Disk TARGET: Obtained from the FQDD field (Section 16.9)

An example of Disk.Bay.2:Enclosure.Internal.0-0:RAID.Slot.1-1 is shown below.

EXAMPLE:

winrm invoke ConvertToNonRAID
"cimv2/root/dcim/DCIM_RAIDService?SystemCreationClassName=DCIM_ComputerSystem+CreationClassName=DCIM_RAIDService+SystemName=DCIM:ComputerSystem+Name=DCIM:RAIDService"
@{PDArray="Disk.Bay.2:Enclosure.Internal.0-0:RAID.Slot.1-1"}
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman
-SkipCNcheck -SkipCAcheck -encoding:utf8 -a:basic -format:pretty
```

OUTPUT:

ConvertToNonRAID_OUTPUT
RebootRequired = 1
ReturnValue = 0

17 Managing BIOS Configuration

This feature provides the ability to get and set any configurable BIOS attributes that are exposed in BIOS UEFI HII. The BIOS Management Profile extends the management capabilities of referencing profiles by adding the capability to represent and configure BIOS attributes, such as a Network Controller or IDE Controller.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

17.1 Listing the BIOS Inventory-Enumeration Class

The BIOS Inventory contains the following attributes: DCIM_BIOSEnumeration (17.1), DCIM_BIOSInteger (17.5), DCIM_BIOSString (17.6), and DCIM_BIOSPassword (17.10).
Enumerating the `BIOSEnumeration` Class will display all BIOS attributes in a computer system.

Enumerate `BIOSEnumeration` with the following parameters and syntax:

**EXAMPLE:**

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BIOSEnumeration
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

```
DCIM_BIOSEnumeration
  AttributeName = NumLock
  CurrentValue = On
  DefaultValue = null
  FQDD = BIOS.Setup.1-1
  InstanceID = BIOS.Setup.1-1:NumLock
  IsReadOnly = false
  PendingValue = null
  PossibleValues = On, Off

DCIM_BIOSEnumeration
  AttributeName = ReportKbdErr
  CurrentValue = Report
  DefaultValue = null
  FQDD = BIOS.Setup.1-1
  InstanceID = BIOS.Setup.1-1:ReportKbdErr
  IsReadOnly = false
  PendingValue = null
  PossibleValues = Report, NoReport

DCIM_BIOSEnumeration
  AttributeName = BootMode
  CurrentValue = Bios
  DefaultValue = null
  FQDD = BIOS.Setup.1-1
  InstanceID = BIOS.Setup.1-1:BootMode
  IsReadOnly = false
  PendingValue = null
  PossibleValues = Bios, Uefi.

DCIM_BIOSEnumeration
  AttributeName = BootSeqRetry
  CurrentValue = Disabled
  DefaultValue = null
  FQDD = BIOS.Setup.1-1
  InstanceID = BIOS.Setup.1-1:BootSeqRetry
  IsReadOnly = false
  PendingValue = null
  PossibleValues = Disabled, Enabled
```

The ‘get’ instance method in Section 17.2 will use this `InstanceId` as input.

The ‘set attribute’ method in Section 17.3 will use the `AttributeName` and `PossibleValues` fields as input.

The ‘set attributes’ method in Section 17.4 will use the `AttributeName` and `PossibleValues` fields as input.
17.2 Getting a BIOS Enumeration Instance

Getting one particular instance of the BIOSEnumeration, instead of all instances as shown in Section 17.1, is shown below.

Get a BIOSEnumeration instance with the following parameters and syntax:

\[
\text{[INSTANCEID]}: \text{This is obtained from the enumeration in Section 17.1, which shows an example using BIOS.Setup.1-1:NumLock as an instanceID}
\]

\[\text{EXAMPLE}:\]

\[
\text{winrm g http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BIOSEnumeration}
\]

\[
\text {?InstanceID}=[[\text{INSTANCE ID}]]
\]

\[
\text {-u:[\text{USER}] -p:[\text{PASSWORD}]}\]

\[
\text {-r://[\text{IPADDRESS}]/wsman -SkipCNcheck -SkipCAcheck}
\]

\[
\text {-encoding:utf8 -a:basic}
\]

\[\text{OUTPUT}:\]

\[
\text{DCIM_BIOSEnumeration}
\]

\[
\text{  AttributeName = NumLock}
\]

\[
\text{  CurrentValue = On}
\]

\[
\text{  DefaultValue = null}
\]

\[
\text{  FQDD = BIOS.Setup.1-1}
\]

\[
\text{  InstanceID = BIOS.Setup.1-1:NumLock}
\]

\[
\text{  IsReadOnly = false}
\]

\[
\text{  PendingValue = null}
\]

\[
\text{  PossibleValues = On, Off}
\]

17.3 Changing the BIOS BootMode-SetAttribute()

The SetAttribute() method can be used to apply changes to setting the BootMode configuration to a given instance.

Invoke SetAttribute() with the following parameters (from Section 17.1) and syntax:

\[
\text{TARGET}: \text{Obtained from the InstanceID field}
\]

\[
\text{AttributeName}: \text{Obtained from the AttributeName field}
\]

\[
\text{AttributeValue}: \text{Obtained from the PossibleValues field}
\]

\[\text{EXAMPLE}:\]

\[
\text{winrm i SetAttribute http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BIOSService}
\]

\[
\text {?SystemCreationClassName=DCIM_ComputerSystem +CreationClassName=DCIM_BIOSService}
\]
17.4 Setting Multiple BIOS BootMode Parameters

Users can find and set multiple BIOS attributes associated with a specific device using the `SetAttributes()` method. This example illustrates how to set the `BiosMode` and `BootSeqRetry` parameters.

Invoke `SetAttributes()` with the following parameters (from Section 17.1) and syntax:

**TARGET:** Obtained from the `InstanceID` field

**AttributeName:** Obtained from the `AttributeName` field

**AttributeValue:** Obtained from the `PossibleValues` field

**EXAMPLE:**

```
winrm i SetAttributes http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BIOSService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_BIOSService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:BIOSService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:SetAttributes_BIOS.xml
```

The input file `SetAttributes_BIOS.xml` is shown below:
  <p:Target>BIOS.Setup.1-1</p:Target>
  <p:AttributeName>BootMode</p:AttributeName>
  <p:AttributeValue>Bios</p:AttributeValue>
  <p:AttributeName>BootSeqRetry</p:AttributeName>
  <p:AttributeValue>Disabled</p:AttributeValue>
</p:SetAttributes_INPUT>

**OUTPUT:**

SetAttribute_OUTPUT

- Message = The command was successful
- MessageID = BIOS001
- RebootRequired = Yes
- ReturnValue = 0
- SetResult = Set PendingValue

### 17.5 Listing the BIOS Inventory-Integer Class

Enumerate *BIOSInteger* with the following parameters and syntax:

**EXAMPLE:**

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BIOSInteger
-u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic
```

**OUTPUT:**

DCIM_BIOSInteger

- AttributeName = AcPwrRcvryUserDelay
- CurrentValue = 0
- DefaultValue = null
- FQDD = BIOS.Setup.1-1
- InstanceID = BIOS.Setup.1-1:AcPwrRcvryUserDelay
- IsReadOnly = true
- LowerBound = 30
- PendingValue = null
- UpperBound = 240

### 17.6 Listing the BIOS Inventory-String Class

Enumerate *BIOSString* with the following parameters and syntax:

**EXAMPLE:**

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BIOSString
-u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic
```
17.7 Applying the Pending Values for BIOS & Boot - CreateTargetedConfigJob()

This method is called to apply the pending values created by the SetAttribute(), SetAttributes(),
ChangeBootOrderByInstanceID(), and ChangeBootSourceState() methods. The system will
automatically reboot depending on the ScheduledStartTime selected. Using the
CreateTargetedConfigJob() jobID output with the job control section can be used to obtain its status.

Invoke CreateTargetedConfigJob() with the following parameters and syntax:

**TARGET:** This Parameter is the FQDD of the BIOSAttribute instances, obtained from the
InstanceID field in Section 17.1

**RebootJobType:** There are three options for rebooting the system.

1 = PowerCycle
2 = Graceful Reboot without forced shutdown
3 = Graceful reboot with forced shutdown

Note: When a user does not want to set a reboot type when creating a target job, users should comment out the RebootJobType in the input xml. User should not enter “0” or give no parameter at all in the input xml.

EXAMPLE:

winrm i CreateTargetedConfigJob
http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BIOSService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM:ComputerSystem
+Name=DCIM:BIOSService
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAncheck
-encoding:utf-8 -a:basic -file:CreateTargetedConfigJob_BIOS.xml

The input file CreateTargetedConfigJob_BIOS.xml is shown below:

```xml
  <p:Target>BIOS.Setup.1-1</p:Target>
  <p:RebootJobType>2</p:RebootJobType>
  <p:ScheduledStartTime>TIME_NOW</p:ScheduledStartTime>
  <p:UntilTime>20111111111111</p:UntilTime>
</p:CreateTargetedConfigJob_INPUT>
```

OUTPUT:

When this method is executed, a jobid or an error message is returned. The status of this jobid can be checked within the job control provider in Section 10.

CreateTargetedConfigJob_OUTPUT
Job
  Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
SelectorSet
  Selector: InstanceID = JID_001269609760, __cimnamespace = root/dcim
ReturnValue = 4096

17.8 Deleting the Pending Values for BIOS & Boot-DeletePendingConfiguration()
This method is called to cancel the pending values created by the SetAttribute() and SetAttributes() methods. The DeletePendingConfiguration() method cancels the pending configuration changes made before the configuration job is created with CreateTargetedConfigJob(). This method only operates on
the pending changes prior to `CreateTargetedConfigJob()` being called. After the configuration job is created, the pending changes can only be canceled by calling `DeleteJobQueue()` in the Job Control profile.

Invoke `CreateTargetedConfigJob()` with the following parameters and syntax:

```
Target:  This parameter is the FQDD of the BIOSAttribute instances (from Section 17.1)
```

**EXAMPLE:**

```
winrm i DeletePendingConfiguration
http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BIOSService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_BIOSService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:BIOSService
-u:[USER] -p:[PASSWORD]
-:https://[IPADDRESS]/wsman -skipCNcheck -skipCAcheck
-encoding:utf-8 -a:basic -file:DeletePendingConfiguration_BIOS.xml
```

The input file `DeletePendingConfiguration_BIOS.xml` is shown below:

```
  <p:Target>BIOS.Setup.1-1</p:Target>
</p:DeletePendingConfiguration_INPUT>
```

**OUTPUT:**

```
DeletePendingConfiguration_OUTPUT
  Message = The command was successful
  MessageID = BIOS001
  ReturnValue = 0
  ReturnValue = 4096
```

### 17.9 Managing BIOS Passwords

The `ChangePassword()` method is used to set the BIOS passwords. The user can either set, change or delete the BIOS system or setup password. Setting the BIOS password is performed in several stages as described in the following sections.

#### 17.9.1 Setting the BIOS Password

The following example sets the BIOS system password to “NEW_PASSWORD”. Three instances of XML are shown below to demonstrate the following scenarios:

- No BIOS password is set
- Changing an existing BIOS password
- Deleting an existing BIOS password

Invoke `ChangePassword()` method with the following parameters:
Target - Obtained from any BIOS enumerate WSMAN command
PasswordType - Either 1 for system or 2 for setup
OldPassword - Reference following XML case A), B) or C)
NewPassword - Reference following XML case A), B) or C)

EXAMPLE:

winrm i ChangePassword http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_BIOSService?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_BIOSService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:BIOSService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:change_bios_password.xml

The input file change_bios_password.xml is shown below:

- No BIOS password is set: The OldPassword parameter is not required. It may be set to “null” or left blank as shown below.
- Changing an existing BIOS password: Both the OldPassword and NewPassword parameters are required. NOTE: Entering only the NewPassword parameter indicates a “pass” in the setting and creating a new job, however the job fails.
- Deleting an existing BIOS password: The OldPassword parameter is required. The NewPassword parameter may be set to “null”, set to blank, or omitted completely.

  <Target>BIOS.Setup.1-1</Target>
  <PasswordType>1</PasswordType>
  <OldPassword></OldPassword>
  <NewPassword>NEW_PASSWORD</NewPassword>
</ChangePassword_INPUT>

OUTPUT:
Either of the following may result:

ChangePassword_OUTPUT
  Message = BIOS does not support Change Password feature
  MessageID = BIOS019
  ReturnValue = 2

ChangePassword_OUTPUT
  Message = The command was successful
  MessageID = BIOS001
17.9.2 Create Target Configuration Job

Create a configuration job as shown in Section 17.7.

17.9.3 Monitor Set BIOS Password Status

To monitor the job status for setting the BIOS password, get the instance of the corresponding job as described within the job control provider in Section 10.

Replace [INSTANCE ID] with the actual jobid from Section 17.9.1.

EXAMPLE:

```
winrm get http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LifecycleJob
?InstanceID=[INSTANCE ID]
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman:443 -SkipCNCheck -SkipCACHCheck
-a:basic -encoding:utf-8
```

OUTPUT:

```
DCIM_LifecycleJob
InstanceID = JID_00129609760
JobStartTime = 0000010000000000
JobStatus = Scheduled
JobUntilTime = TIME_NA
Message = Task successfully scheduled
MessageID = JCP001
Name = ConfigBIOS:BIOS.Setup.1-1
PercentComplete = NA
```

The status may be one of the following:

- **Ready for execution** - Job is created, but waiting for scheduled start time to pass to schedule the job
- **Scheduled** - Job is scheduled and ready for system reboot to execute the job
- **Failed** - Problem with setting the BIOS password, check message for more information
  - **Completed** - Setting the BIOS password completed with no issues

17.10 Listing the BIOS Inventory-Password Class

Enumerate *BIOSPassword* with the following parameters and syntax:

EXAMPLE:
winrm e cimv2/root/dcim/DCIM_BIOSPassword
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic

OUTPUT:

DCIM_BIOSPassword
   AttributeDisplayName = System Password
   AttributeName = SysPassword
   Dependency = <Dep><AttrLev Op="OR"> <ROIf Name="PasswordStatus"> Locked </ROIf></AttrLev></Dep>
   DisplayOrder = 1402
   FQDD = BIOS.Setup.1-1
   GroupDisplayName = System Security
   GroupID = SysSecurity
   InstanceID = BIOS.Setup.1-1:SysPassword
   IsReadOnly = false
   IsSet = false
   MaxLength = 32
   MinLength = 0
   PasswordState = 3
   PendingValue = null
   ValueExpression = ^[\]0-9a-z "+,-/,;\[\]0,32]$

DCIM_BIOSPassword
   AttributeDisplayName = Setup Password
   AttributeName = SetupPassword
   Dependency = null
   DisplayOrder = 1403
   FQDD = BIOS.Setup.1-1
   GroupDisplayName = System Security
   GroupID = SysSecurity
   InstanceID = BIOS.Setup.1-1:SetupPassword
   IsReadOnly = false
   IsSet = false
   MaxLength = 32
   MinLength = 0
   PasswordState = 3
   PendingValue = null
   ValueExpression = ^[\]0-9a-z "+,-/,;\[\]0,32]$

18  Exporting and Importing Server Profile

Use this feature to back up and restore host server profile. You can take a backup of current system configuration that is stored in a backup image file. Use Restore at anytime to put the system to pre-backup state.

Profile and Associated MOFs:
18.1 Exporting Server Profile

To backup host system server profile, invoke the `BackupImage()` method in the class `DCIM_LCService`. Backup feature gathers system information, firmware images, hardware configuration, Lifecycle Controller, iDRAC firmware, and configuration and stores the information in a file. You can save the file on either iDRAC vFlash SD card or network share.

- **[IP ADDRESS]**: This is the IP address of the file server.
- **[DRIVESHARE]**: This is the directory path for the image.
- **[USERNAME]**: This is the username to the file share.
- **[PASSWORD]**: This is the password to the file share.
- **[IMAGENAME]**: This is the desired name of the image.
- **[PASSPHRASE]**: This can be used to password protect NFS and CIFS images.

For NFS and CIFS shares, the entire “Passphrase=’[PASSPHRASE]’;” argument is optional. Note: To restore this backup file, the same passphrase must passed as an argument for the operation to be successful.

The following examples back up the server profile and execute it immediately, using the `TIME_NOW` parameter.

18.1.1 Exporting Server Profile to iDRAC vFlash Card-BackupImage()

**iDRAC vFlash Card:**

*ShareType* is “4”.

**EXAMPLE**:

```
winrm i BackupImage http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService ?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]:443/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic @([IPAddress]="[IP ADDRESS]"; ShareType="4"; ScheduledStartTime="TIME_NOW")
```

18.1.2 Exporting Server Profile to NFS Share-BackupImage()
NFS Share:

`ShareType` is “0”. The entire “Passphrase=“passphrase”;” argument is optional.

**EXAMPLE**:

```bash
winrm i BackupImage http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
  ?SystemCreationClassName=DCIM_ComputerSystem +CreationClassName=DCIM_LCService +SystemName=DCIM:ComputerSystem
  +Name=DCIM:LCService
  -u:[USER] -p:[PASSWORD]
  -r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
  -encoding:utf-8 -a:basic @@[IPADDRESS]="[IP ADDRESS]";
  ShareName="/[DRIVESHARE]";ShareType="0";ImageName="[IMAGENAME]";
  Username="[USERNAME]";Password="[PASSWORD]";Passphrase="[PASSPHRASE]";
  ScheduledStartTime="TIME_NOW"}
```

**NOTE:** The ShareName field should only be the folder exposed by the system to the network. Any sub folder information should be attached to the ImageName field. Otherwise, there can be connection issues when trying to locate/create the backup file.

Correct Example: `ShareName="/folder1";ImageName="subfolder/image_name"

Incorrect Example: `ShareName="/folder1/subfolder";ImageName="image_name"

### 18.1.3 Exporting Server Profile to CIFS Share-BackupImage()

CIFS Share:

`ShareType` is “2”. The entire “Passphrase=“passphrase”;” argument is optional.

**EXAMPLE**:

```bash
winrm i BackupImage http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
  ?SystemCreationClassName=DCIM_ComputerSystem +CreationClassName=DCIM_LCService +SystemName=DCIM:ComputerSystem
  +Name=DCIM:LCService
  -u:[USER] -p:[PASSWORD]
  -r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
  -encoding:utf-8 -a:basic@[IPADDRESS]="[IP ADDRESS]";
  ShareName="/[DRIVESHARE]";ShareType="2";ImageName="[IMAGENAME]";
  Username="[USERNAME]";Password="[PASSWORD]";Passphrase="[PASSPHRASE]";
  ScheduledStartTime="TIME_NOW"}
```

**NOTE:** The ShareName field should only be the folder exposed by the system to the network. Any sub folder information should be attached to the ImageName field. Otherwise, there can be connection issues when trying to locate/create the backup file.

Correct Example: `ShareName="/folder1";ImageName="subfolder/image_name"`
In-Correct Example:  ShareName="/folder1/subfolder";ImageName="image_name"

**OUTPUT:**

BackupImage_OUTPUT

Job


ReferenceParameters

  ResourceURI = http://schemas.dell.com/wbem/wscim/1/cimschema/2/DCIM_LifecycleJob

SelectorSet

  Selector: InstanceID = JID_001293618214, __cimnamespace = root/dcim

ReturnValue = 4096

The response contains a reference to the job class that will provide the status of the operation.

The return value is 4096 which indicates that the method operation is not yet complete.

**18.1.4 Monitoring Export status**

Backup process may take up to 30 minutes depending on host system configuration. To monitor the backup status, get the instance of the corresponding job.

Replace [INSTANCE ID] with the actual jobid from Section 18.1.1, 18.1.2, or 18.1.3.

**EXAMPLE:**

```
-u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman:443 -SkipCNCheck -SkipCACheck -a:basic -encoding:utf-8
```

**OUTPUT:**

```
DCIM_LifecycleJob

  InstanceID = JID_001293618214
  JobStartTime = 00000101000000
  JobStatus = Backup In Progress
  JobUntilTime = TIME_NA
  Message = Collecting Lifecycle Controller Firmware images
  MessageID = BAR063
  Name = Backup:Image
  PercentComplete = 50
```

The status may be one of the following:

- **Ready for Backup** - Request is received
- **Backup In Progress** - Backup process is currently in process
- **Failed** - Problem with the backup process, check message for more information
- **Completed** - Backup process is complete with no issues
18.2 Importing Server Profile

To restore host system server profile, invoke the RestoreImage() method in the class DCIM_LCService. Restore process restores the system information, firmware images, hardware configuration, Lifecycle Controller, iDRAC firmware, and configuration from the backup image file located on either iDRAC vFlash SD card or network share.

[IP ADDRESS]: This is the IP address of the file server.
[DRIVESHARE]: This is the directory path for the image.
[USERNAME]: This is the username to the file share.
[PASSWORD]: This is the password to the file share.
[IMAGENAME]: This is the desired name of the image.
[PASSPHRASE]: This can be used to password protect NFS and CIFS images.

For NFS and CIFS shares, the entire “Passphrase=“[PASSPHRASE]”;” argument is only required when the backup image used a passphrase.

The following examples restore the server profile and execute it immediately, using the TIME_NOW parameter.

18.2.1 Importing Server Profile from iDRAC vFlash Card-RestoreImage()

iDRAC vFlash Card:

ShareType is “4”.

winrm i RestoreImage http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService ?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_LCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:LCService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic @
{IPAddress=“[IP ADDRESS]”;ShareType=“4”;ScheduledStartTime=“TIME_NOW”}

18.2.2 Importing Server Profile from NFS share-RestoreImage()

NFS Share:

ShareType is “0”.

EXAMPLE:
18.2.3 Importing Server Profile from CIFS share-RestoreImage()

CIFS Share:

ShareType is “2”.

NOTE: The ShareName field should only be the folder exposed by the system to the network. Any sub folder information should be attached to the ImageName field. Otherwise, there can be connection issues when trying to locate/create the backup file.

Correct Example:     ShareName="/folder1";ImageName="subfolder/image_name"

In-Correct Example:  ShareName="/folder1/subfolder";ImageName="image_name"
OUTPUT:

RestoreImage_OUTPUT
  Job
    Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
ReferenceParameters
SelectorSet
  Selector: InstanceID = JID_001293618214, __cimnamespace = root/dcim
ReturnValue = 4096

The response contains a reference to the job class that will provide the status of the operation. The return value is 4096 which indicates that the method operation is not yet complete.

18.2.4 Monitoring Import Status

Restore process may take up to 60 minutes depending on host system configuration. To monitor the backup status, get the instance of the corresponding job.

Replace [INSTANCE ID] with the actual jobid from Section 18.2.1, 18.2.2, or 18.2.3.

EXAMPLE:

winrm get http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LifecycleJob
?InstanceID=[INSTANCE ID]
-u: [USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -SkipCNCheck -SkipCACheck
-a:basic -encoding:utf-8

OUTPUT:

DCIM_LifecycleJob
  InstanceID = JID_001293618214
  JobStartTime = 00000101000000
  JobStatus = Restore In Progress
  JobUntilTime = TIME_NA
  Message = Restoring Lifecycle Controller Firmware images
  MessageID = BAR081
  Name = Restore:Image
  PercentComplete = 20

The status may be one of the following:

- **Ready for Restore** - Request has been received
- **Restore In Progress** - Restore process is currently in process
- **Failed** - Problem with the restore process, check message for more information
• Completed: Restore process has completed with no issues

19 iDRAC Configuration

This feature provides the ability to remotely list, get, and set the attributes on various monolithic and modular servers for the three Dell iDRAC classes through the command line.

- DCIM_iDRACCardEnumeration (19.1)
- DCIM_iDRACCardInteger (19.4)
- DCIM_iDRACCardString (19.6)

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

19.1 Listing the iDRAC Card Inventory-Enumeration Class

Enumerate the iDRACCardEnumeration class to list all the enumerate, integer, and string type iDRAC attributes.

Enumerate the iDRACCardEnumeration class with the following parameters and syntax:

**EXAMPLE:**

```
winrme http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardEnumeration
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

```
DCIM_iDRACCardEnumeration
    AttributeDisplayName = Nic Enable
    AttributeName = Enable
    CurrentValue = Enabled
    DefaultValue = Enabled
    Dependency = null
    DisplayOrder = 0
    FQDD = iDRAC.Embedded.1
    GroupDisplayName = NIC
    GroupID = NIC.1
    InstanceID = iDRAC.Embedded.1#NIC.1#Enable
    IsReadOnly = false
    PossibleValues = Disabled, Enabled
```

DCIM_iDRACCardEnumeration
AttributeDisplayName = Virtual Media Attached
AttributeName = Attached
CurrentValue = Detached
DefaultValue = Detached
Dependency = null
DisplayOrder = 0
FQDD = iDRAC.Embedded.1
GroupDisplayName = VirtualMedia
GroupID = VirtualMedia.1
InstanceID = iDRAC.Embedded.1#VirtualMedia.1#Attached
IsReadOnly = false
PossibleValues = Detached, Attached, Autoattach

DCIM_iDRACCardEnumeration
AttributeDisplayName = IPv4 Enable
AttributeName = Enable
CurrentValue = Enabled
DefaultValue = Enabled
Dependency = null
DisplayOrder = 0
FQDD = iDRAC.Embedded.1
GroupDisplayName = IPv4
GroupID = IPv4.1
InstanceID = iDRAC.Embedded.1#IPv4.1#Enable
IsReadOnly = false
PossibleValues = Disabled, Enabled

19.2 Getting an iDRAC Card Enumeration Instance

Use the following example to get an instance of the DCIM_iDRACCardEnumeration class instead of all the instances as shown in Section 19.1.

Get an iDRACCardEnumeration instance with the following parameters and syntax:

[INSTANCEID]: This is obtained from the enumeration in Section 19.1, which shows an example using iDRAC.Embedded.1#NIC.1#Enable as an instanceID.
EXAMPLE:

```
winrm g http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardEnumeration
?InstanceID=[INSTANCE ID]
-u:[USER] -p:[PASSWORD]
-r://[IPADDRESS]/wsman:443
-SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic
```

OUTPUT:

```
DCIM_iDRACCardEnumeration
    AttributeDisplayName = Nic Enable
    AttributeName = Enable
    CurrentValue = Enabled
    DefaultValue = Enabled
    Dependency = null
    DisplayOrder = 0
    FQDD = iDRAC.Embedded.1
    GroupDisplayName = NIC
    GroupID = NIC.1
    InstanceID = iDRAC.Embedded.1#NIC.1#Enable
    IsReadOnly = false
    PossibleValues = Disabled, Enabled
```

19.3 Listing the iDRAC Card Inventory-Enumeration Class using groupID

Enumerate the DCIM_iDRACCardEnumeration class to list all the enumerate type iDRAC attributes using the group IDs of these groups: NIC, VirtualMedia, IPv4, and Users. To retrieve the attributes of the groups, set the GroupID to one of the following: NIC, VirtualMedia, IPv4, or Users.

Enumerate the `iDRACCardEnumeration` class using the following parameters and syntax:

EXAMPLE:

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardEnumeration
-u:[USER] -p:[PASSWORD]
-r://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
-dialect:http://schemas.microsoft.com/wbem/wsman/1/WQL
-filter: select * from DCIM_iDRACCardEnumeration WHERE GroupID="NIC.1"
```

The possible inputs for GroupID are:

NIC.1
VirtualMedia.1

IPv4.1

Users.3

OUTPUT:

```
DCIM_iDRACCardEnumeration
  AttributeDisplayName = Nic Enable
  AttributeName = Enable
 CurrentValue = Enabled
  DefaultValue = Enabled
  Dependency = null
  DisplayOrder = 0
  FQDD = iDRAC.Embedded.1
  GroupDisplayName = NIC
  GroupID = NIC.1
  InstanceID = iDRAC.Embedded.1#NIC.1#Enable
  IsReadOnly = false
  PossibleValues = Disabled, Enabled

DCIM_iDRACCardEnumeration
  AttributeDisplayName = Virtual Media Attached
  AttributeName = Attached
 CurrentValue = Attached
  DefaultValue = Detached
  Dependency = null
  DisplayOrder = 0
  FQDD = iDRAC.Embedded.1
  GroupDisplayName = VirtualMedia
  GroupID = VirtualMedia.1
  InstanceID = iDRAC.Embedded.1#VirtualMedia.1#Attached
  IsReadOnly = false
  PossibleValues = Detached, Attached, Autoattach

DCIM_iDRACCardEnumeration
  AttributeDisplayName = IPv4 Enable
  AttributeName = Enable
 CurrentValue = Enabled
  DefaultValue = Enabled
  Dependency = null
  DisplayOrder = 0
  FQDD = iDRAC.Embedded.1
  GroupDisplayName = IPv4
  GroupID = IPv4.1
  InstanceID = iDRAC.Embedded.1#IPv4.1#Enable
  IsReadOnly = false
  PossibleValues = Disabled, Enabled

DCIM_iDRACCardEnumeration
  AttributeDisplayName = User Admin IPMI LAN Privilege
  AttributeName = IpmiLanPrivilege
```
CurrentValue = Administrator
DefaultValue = NoAccess
Dependency = null
DisplayOrder = 0
FQDD = iDRAC.Embedded.1
GroupDisplayName = Users
GroupId = Users.3
InstanceID = iDRAC.Embedded.1#Users.3#IpmiLanPrivilege
IsReadOnly = false
PossibleValues = User, Operator, Administrator, NoAccess

19.4 Applying the Attributes and Polling Job Completion

19.4.1 Changing iDRAC Values-ApplyAttributes() (Immediate)

Invoke the ApplyAttributes() method on the DCIM_iDRACCardService class to set or change the value of one or many enumerate type attributes. This method takes an xml file as input. The changes to the attributes are defined in this xml file. This method returns a JobID that is used as input in the next section (Section 19.3.2).

Invoke ApplyAttributes() method with the following parameters and syntax:

EXAMPLE:

```
winrm i ApplyAttributes http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardService
+SystemCreationClassName=DCIM_ComputerSystem +CreationClassName=DCIM_iDRACCardService +SystemName=DCIM:ComputerSystem +Name=DCIM:iDRACCardService -u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -file:DRACService_SetAttribute_group_enumerate.xml
```

The input file DRACService_SetAttribute_group_enumerate.xml is shown below.

```xml
  <p:Target>iDRAC.Embedded.1</p:Target>
  <p:AttributeName>NIC.1#Enable</p:AttributeName>
  <p:AttributeValue>Enabled</p:AttributeValue>
  <p:AttributeName>NIC.1#Selection</p:AttributeName>
  <p:AttributeValue>Dedicated</p:AttributeValue>
  <p:AttributeName>NIC.1#Speed</p:AttributeName>
  <p:AttributeValue>100</p:AttributeValue>
  <p:AttributeName>NIC.1#Autoneg</p:AttributeName>
  <p:AttributeValue>Enabled</p:AttributeValue>
  <p:AttributeName>NIC.1#Duplex</p:AttributeName>
  <p:AttributeValue>Full</p:AttributeValue>
  <p:AttributeName>NIC.1#DNSRegister</p:AttributeName>
  <p:AttributeValue>Enabled</p:AttributeValue>
  <p:AttributeName>NIC.1#DNSDomainNameFromDHCP</p:AttributeName>
</p:ApplyAttributes_INPUT>
```
<p:AttributeValue>Enabled</p:AttributeValue>
<p:AttributeName>NIC.1#VlanEnable</p:AttributeName>
<p:AttributeValue>Disabled</p:AttributeValue>
<p:AttributeName>VirtualMedia.1#Attached</p:AttributeName>
<p:AttributeValue>Dettached</p:AttributeValue>
<p:AttributeName>IPv4.1#Enable</p:AttributeName>
<p:AttributeValue>Enabled</p:AttributeValue>
<p:AttributeName>IPv4.1#DHCPEnable</p:AttributeName>
<p:AttributeValue>Enabled</p:AttributeValue>
<p:AttributeName>IPv4.1#DNSFromDHCP</p:AttributeName>
<p:AttributeValue>Enabled</p:AttributeValue>
<p:AttributeName>Users.3#Enable</p:AttributeName>
<p:AttributeValue>Enabled</p:AttributeValue>
<p:AttributeName>Users.3#IpmiLanPrivilege</p:AttributeName>
<p:AttributeValue>Administrator</p:AttributeValue>
<p:AttributeName>Users.16#IpmiLanPrivilege</p:AttributeName>
<p:AttributeValue>Administrator</p:AttributeValue>
<p:AttributeName>Users.3#IpmiSerialPrivilege</p:AttributeName>
<p:AttributeValue>Administrator</p:AttributeValue>
<p:AttributeName>Users.16#IpmiSerialPrivilege</p:AttributeName>
<p:AttributeValue>Administrator</p:AttributeValue>
</p:ApplyAttributes_INPUT>

**OUTPUT:**

ApplyAttributes_OUTPUT
  Job
    Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    ReferenceParameters
      SelectorSet
        Selector: InstanceID = JID_001293705757, __cimnamespace = root/dcim
    ReturnValue = 4096

19.4.2 Polling Job Completion

Use the Get() command to check the progress of the ApplyAttributes() method. It polls for job completion. This method takes the InstanceID from the previous section (19.3.1) as input. The JobStatus value is either “Successful” or “Failed”. If the job failed, the Message value contains more detailed error information on the cause of the failure.

Run the Get() command on DCIM_LifecycleJob with the following parameters and syntax:

**EXAMPLE:**

205
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCACheck
-encoding:utf-8 -a:basic

The input parameter is the InstanceID from the output of the ApplyAttributes() method. An example InstanceID is as follows: InstanceID = JID_001293705757

OUTPUT:

DCIM_LifecycleJob
    InstanceID = JID_001293705757
    JobStartTime = TIME_NA
    JobStatus = Completed
    JobUntilTime = TIME_NA
    Message = NA
    MessageID = NA
    Name = iDRACConfig:iDRAC.Embedded.1
    PercentComplete = 100

19.4.3 Set Attribute Verification
To verify the changes made to the attributes, enumerate the DCIM_iDRACCardEnumeration class. For more information, see Section 19.1.

OUTPUT #2:

DCIM_iDRACCardEnumeration
    AttributeDisplayName = Nic Enable
    AttributeName = Enable
    CurrentValue = Enabled
    DefaultValue = Enabled
    Dependency = null
    DisplayOrder = 0
    FQDD = iDRAC.Embedded.1
    GroupDisplayName = NIC
    GroupID = NIC.1
    InstanceID = iDRAC.Embedded.1#NIC.1#Enable
    IsReadOnly = false
    PossibleValues = Disabled, Enabled

DCIM_iDRACCardEnumeration
    AttributeDisplayName = Virtual Media Attached
    AttributeName = Attached
    CurrentValue = Attached
    DefaultValue = Detached
    Dependency = null
    DisplayOrder = 0
    FQDD = iDRAC.Embedded.1
    GroupDisplayName = VirtualMedia
    GroupID = VirtualMedia.1
    InstanceID = iDRAC.Embedded.1#VirtualMedia.1#Attached
IsReadOnly = false
PossibleValues = Detached, Attached, Autoattach

DCIM_iDRACCardEnumeration
AttributeDisplayName = IPv4 Enable
AttributeName = Enable
CurrentValue = Enabled
DefaultValue = Enabled
Dependency = null
DisplayOrder = 0
FQDD = iDRAC.Embedded.1
GroupDisplayName = IPv4
GroupID = IPv4.1
InstanceID = iDRAC.Embedded.1#IPv4.1#Enable
IsReadOnly = false
PossibleValues = Disabled, Enabled

DCIM_iDRACCardEnumeration
AttributeDisplayName = User Admin IPMI LAN Privilege
AttributeName = IpmiLanPrivilege
CurrentValue = Administrator
DefaultValue = NoAccess
Dependency = null
DisplayOrder = 0
FQDD = iDRAC.Embedded.1
GroupDisplayName = Users
GroupID = Users.3
InstanceID = iDRAC.Embedded.1#Users.3#IpmiLanPrivilege
IsReadOnly = false
PossibleValues = User, Operator, Administrator, NoAccess

19.5 Listing the iDRAC Card Inventory-Integer Class

Enumerate the DCIM_iDRACCardInteger class to list all the integer type iDRAC attributes.

Enumerate the DCIM_iDRACCardInteger class with the following parameters and syntax:

EXAMPLE:

`winrm e http://schemas.dmtf.org/wbem/wsci/1/cim-schema/2/root/dcim/DCIM_iDRACCardInteger
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCACheck
-encoding:utf-8 -a:basic`

OUTPUT:

DCIM_iDRACCardInteger
AttributeDisplayName = VLan Priority
AttributeName = VLanPriority
CurrentValue = 0
DefaultValue = 0
Dependency = null
DisplayOrder = 0
FQDD = iDRAC.Embedded.1
GroupDisplayName = NIC
GroupID = NIC.1
InstanceID = iDRAC.Embedded.1#NIC.1#VlanPriority
IsReadOnly = false
LowerBound = 0
UpperBound = 7

DCIM_iDRACCardInteger
   AttributeDisplayName = User Admin Privilege
   AttributeName = Privilege
   CurrentValue = 511
   DefaultValue = 0
   Dependency = null
   DisplayOrder = 0
   FQDD = iDRAC.Embedded.1
   GroupDisplayName = Users
   GroupID = Users.3
   InstanceID = iDRAC.Embedded.1#Users.3#Privilege
   IsReadOnly = false
   LowerBound = 0
   UpperBound = 511

19.6  Listing the iDRAC Card Inventory-Integer Class using groupId

Enumerate the DCIM_iDRACCardInteger class to list all the integer type iDRAC attributes using the group IDs of these groups: NIC and Users. To retrieve the attributes of the groups, set the GroupID to one of the following: NIC or Users.

All the iDRAC attributes of type integer that are part of a given Group (NIC and Users) are retrieved. In order to do this, “GroupID” needs to be set to one of the following: NIC or Users.

Enumerate the iDRACCardInteger class with the following parameters and syntax:

EXAMPLE:

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardInteger
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
-dialect:http://schemas.microsoft.com/wbem/wsman/1/WQL
-filter:`select * from DCIM_iDRACCardInteger WHERE GroupID='NIC.1'`
```

The possible inputs for GroupID are:

NIC.1
Users.3

OUTPUT:
DCIM_iDRACCardInteger
   AttributeDisplayName = VLan Priority
   AttributeName = VLanPriority
   CurrentValue = 1
   DefaultValue = 0
   Dependency = null
   DisplayOrder = 0
   FQDD = iDRAC.Embedded.1
   GroupDisplayName = NIC
   GroupID = NIC.1
   InstanceID = iDRAC.Embedded.1#NIC.1#VLanPriority
   IsReadOnly = false
   LowerBound = 0
   UpperBound = 7

DCIM_iDRACCardInteger
   AttributeDisplayName = User Admin Privilege
   AttributeName = Privilege
   CurrentValue = 511
   DefaultValue = 0
   Dependency = null
   DisplayOrder = 0
   FQDD = iDRAC.Embedded.1
   GroupDisplayName = Users
   GroupID = Users.3
   InstanceID = iDRAC.Embedded.1#Users.3#Privilege
   IsReadOnly = false
   LowerBound = 0
   UpperBound = 511

19.7 Listing the iDRAC Card Inventory-String Class
Enumerate the DCIM_iDRACCardString class to list all the string type iDRAC attributes.

Enumerate the iDRACCardString class with the following parameters and syntax:

**EXAMPLE:**
```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardString
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**
```
DCIM_iDRACCardString
   AttributeDisplayName = DNS RAC Name
   AttributeName = DNSRacName
   CurrentValue = idrac
   DefaultValue
   Dependency = null
   DisplayOrder = 0
```
FQDD = iDRAC.Embedded.1
GroupDisplayName = NIC
GroupID = NIC.1
InstanceID = iDRAC.Embedded.1#NIC.1#DNSRacName
IsReadOnly = false
MaxLength = 63
MinLength = 1

DCIM_iDRACCardString
  AttributeDisplayName = IP Address
  AttributeName = Address
 CurrentValue = 172.27.36.55
  DefaultValue = 192.168.0.120
  Dependency = null
  DisplayOrder = 0
  FQDD = iDRAC.Embedded.1
  GroupDisplayName = IPv4
  GroupID = IPv4.1
  InstanceID = iDRAC.Embedded.1#IPv4.1#Address
  IsReadOnly = false
  MaxLength = 16
  MinLength = 1

DCIM_iDRACCardString
  AttributeDisplayName = User Admin User Name
  AttributeName = UserName
 CurrentValue = dell3
  DefaultValue
  Dependency = null
  DisplayOrder = 0
  FQDD = iDRAC.Embedded.1
  GroupDisplayName = Users
  GroupID = Users.3
  InstanceID = iDRAC.Embedded.1#Users.3#UserName
  IsReadOnly = false
  MaxLength = 16
  MinLength = 1

19.8  Listing the iDRAC Card Inventory-String Class using groupID

Enumerate the DCIM_iDRACCardString class to list all the string type iDRAC attributes using the group IDs of these groups: NIC, IPv4, and Users. To retrieve the attributes of the groups, set the GroupID to one of the following: NIC, IPv4, or Users.

Invoke dracgetgroupid_string with the following parameters and syntax:

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardstring
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
-dialect:http://schemas.microsoft.com/wbem/wsman/1/WQL
-filter: "select * from DCIM_iDRACCardString WHERE GroupID='NIC.1'"

The possible inputs for GroupID are:

NIC.1
IPv4.1
Users.3

OUTPUT:

DCIM_iDRACCardString
  AttributeDisplayName = DNS RAC Name
  AttributeName = DNSRacName
  CurrentValue = IDRAC
  DefaultValue
  Dependency = null
  DisplayOrder = 0
  FQDD = iDRAC.Embedded.1
  GroupDisplayName = NIC
  GroupID = NIC.1
  InstanceID = iDRAC.Embedded.1#NIC.1#DNSRacName
  IsReadOnly = false
  MaxLength = 63
  MinLength = 1

DCIM_iDRACCardString
  AttributeDisplayName = IP Address
  AttributeName = Address
  CurrentValue = 172.27.36.55
  DefaultValue = 192.168.0.120
  Dependency = null
  DisplayOrder = 0
  FQDD = iDRAC.Embedded.1
  GroupDisplayName = IPv4
  GroupID = IPv4.1
  InstanceID = iDRAC.Embedded.1#IPv4.1#Address
  IsReadOnly = false
  MaxLength = 16
  MinLength = 1

DCIM_iDRACCardString
  AttributeDisplayName = User Admin User Name
  AttributeName = UserName
  CurrentValue = dell3
  DefaultValue
  Dependency = null
  DisplayOrder = 0
  FQDD = iDRAC.Embedded.1
  GroupDisplayName = Users
  GroupID = Users.3
  InstanceID = iDRAC.Embedded.1#Users.3#UserName
IsReadOnly = false
MaxLength = 16
MinLength = 1

19.9 Changing the iDRAC IPChange Notification

19.9.1 Getting the Current iDRAC IPChange State

Get the IPChangeNotifyPS attribute from the DCIM_LCAtribute class to display. The CurrentValue field indicates the current status of this attribute.

EXAMPLE:

winrm get http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCAtribute
?InstanceId=DCIM_LCEnumeration:DHS3
-u:[USER] -p:[PASSWORD]
-r:[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic

OUTPUT:

DCIM_LCEnumeration
    AttributeName = IPChangeNotifyPS
    Caption = null
    CurrentValue = Off
    DefaultValue = Off
    Description = null
    ElementName = LC.emb.1
    InstanceID = DCIM_LCEnumeration:DHS3
    IsOrderedList = null
    IsReadOnly = true
    PendingValue = null
    PossibleValues = On, Off
    PossibleValuesDescription = null

19.9.2 Setting the iDRAC IPChange Notification-SetAttribute()

The SetAttribute() method is used to set the attribute IPChangeNotifyPS to “ON” or “OFF”. When set to “ON”, a user notification is sent when the IP address is changed. While set to “OFF”, a user notification is not sent.

Invoke SetAttribute() with the following syntax:

EXAMPLE:

winrm i SetAttribute http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_LCService
The input file `setattribute.xml` is shown below:

```
  <p:AttributeName>IPChangeNotifyPS</p:AttributeName>
  <p:AttributeValue>on</p:AttributeValue>
</p:SetAttribute_INPUT>
```

**OUTPUT:**

```
SetAttribute_OUTPUT
  ReturnValue = 0
```

To verify the changes after setattribute was executed, list the LC attributes as shown in Section 19.8.1.

## 20 Remote Service Status

To get the remote service status, invoke the `GetRemoteServicesAPIStatus()` method in the class `DCIM_LCService`. This method is used to obtain the overall remote services API status that includes both the host system status as well as the Lifecycle Controller (Data Manager included) status. The overall rolled up status shall be reflected in the Status output parameter.

**NOTE:** The LCStatus output parameter value includes the status reported by the DMStatus output parameter in the GetRSSStatus() method. Thus, GetRSSStatus() method invocation is redundant.

Profile and Associated MOFs:

### 20.1 Getting Remote Service Status

**EXAMPLE:**

```
```
GetRemoteServicesAPIStatus_OUTPUT

LCStatus = 0
Message = Lifecycle Controller Remote Services is ready.
MessageID = LC061
ReturnValue = 0
ServerStatus = 2
Status = 0

Details on each output parameter is described below:

<table>
<thead>
<tr>
<th>Output parameter Name</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>0 (Ready)</td>
<td>Lifecycle Controller Remote Services is ready to accept any web services request.</td>
</tr>
<tr>
<td></td>
<td>1 (Not Ready)</td>
<td>Lifecycle Controller Remote Services is currently not ready to accept web services request. This could be because the instrumentation in iDRAC might be reloading /not_ready or server is in POST or performing scheduled provisioning requests or Lifecycle Controller Unified Server Configurator is in use.</td>
</tr>
<tr>
<td>MessageID</td>
<td>LC060</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LC061</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>Lifecycle Controller Remote Services is not ready.</td>
<td>Message for ID LC060</td>
</tr>
<tr>
<td></td>
<td>Lifecycle Controller Remote Services is ready.</td>
<td>Message for ID LC061</td>
</tr>
<tr>
<td>ServerStatus</td>
<td>0 (Powered off)</td>
<td>Server is powered off</td>
</tr>
<tr>
<td></td>
<td>1 (In POST)</td>
<td>Server is performing normal POST operation</td>
</tr>
<tr>
<td></td>
<td>2 (Out of POST)</td>
<td>Server is out of POST</td>
</tr>
<tr>
<td></td>
<td>3 (Collecting System Inventory)</td>
<td>Server is currently executing UEFI Collect System Inventory On Restart application</td>
</tr>
<tr>
<td></td>
<td>4 (Automated Task Execution)</td>
<td>Server is currently executing scheduled jobs using UEFI Automated Task application</td>
</tr>
<tr>
<td>LCStatus</td>
<td>Server status</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0 (Ready)</td>
<td>Lifecycle Controller instrumentation is up to date and enabled</td>
<td></td>
</tr>
<tr>
<td>1 (Not Initialized)</td>
<td>Lifecycle Controller instrumentation is not initialized. The initialization operation may take up to a minute.</td>
<td></td>
</tr>
<tr>
<td>2 (Reloading Data)</td>
<td>Lifecycle Controller instrumentation is currently refreshing its cache because of a recent configuration change. The reloading operation typically takes few seconds and could take up to few minutes to complete.</td>
<td></td>
</tr>
<tr>
<td>3 (Disabled)</td>
<td>Lifecycle Controller is disabled on the server. Lifecycle Controller can be enabled thru Remote Services or F2 iDRAC configuration.</td>
<td></td>
</tr>
<tr>
<td>4 (In Recovery)</td>
<td>Lifecycle Controller is in Recovery mode. Refer to iDRAC users guide on instructions on how to repair Lifecycle Controller.</td>
<td></td>
</tr>
<tr>
<td>5 (In Use)</td>
<td>Lifecycle Controller is being currently used by another process.</td>
<td></td>
</tr>
</tbody>
</table>

20.2 Restarting Remote Service Status

If you continue to get “Not Ready” remote service status, invoke the `DeleteJobQueue()` method with JID_CLEARALL job id to restart the remote service [LC1.5.x ONLY].

**EXAMPLE:**

```
winrm invoke DeleteJobQueue cimv2/root/dcim/DCIM_JobService
?CreationClassName=DCIM_JobService
+Name=JobService
+SystemName=Idrac
+SystemCreationClassName=DCIM_ComputerSystem
@{JobID="JID_CLEARALL" }
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman:443 -SkipCNCheck -SkipCACheck -auth:basic -encoding:utf-8
```

**OUTPUT:**

```
DeleteJobQueue_OUTPUT
  Message = The specified job was deleted
  MessageID = SUP020
  ReturnValue = 0
```
21 System Information

The DCIM System Info Profile describes the properties and interfaces for executing system management tasks related to the management of the host system. The profile standardizes and aggregates the description for the platform’s basic properties into a system view representation and provides static methodology for the clients to query the system views without substantial traversal of the model.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

21.1 Listing the System Inventory-SystemView Class

The system view returns various information about the system, including the currently installed Lifecycle Controller version as shown below.

Enumerate the DCIM_SystemView class with the following parameters and syntax:

EXAMPLE:

```
winrm e cimv2/root/dcim/DCIM_SystemView
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic
```

OUTPUT:

```
DCIM_SystemView
AssetTag
BIOSReleaseDate = 01/09/2012
BIOSVersionString = 0.3.37
BaseBoardChassisSlot = NA
BatteryRollupStatus = 1
BladeGeometry = 4
BoardPartNumber = 0MX4YFX04
BoardSerialNumber = CN13740184000Q
CMCIP = null
CPLDVersion = 1.0.0
CPURollupStatus = 1
ChassisName = Main System Chassis
ChassisServiceTag = 7654321
ChassisSystemHeight = 5
ExpressServiceCode = 15608862073
FQDD = System.Embedded.1
FanRollupStatus = 3
HostName
InstanceID = System.Embedded.1
LastSystemInventoryTime = 20120116145530.000000+000
LastUpdateTime = 20120116124210.000000+000
```
22 Sensor Information

The DCIM Sensors Profile describes the properties and interfaces for executing system management tasks related to the management of sensors within a system.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

22.1 Listing the Sensors Inventory-PSNumericSensor Class

Enumerate the `DCIM_PSNumericSensor` class with the following parameters and syntax:

**EXAMPLE:**

```csharp
LicensingRollupStatus = 1
LifecycleControllerVersion = 2.0.0
Manufacturer = Dell Inc.
MaxCPUSockets = 2
MaxDIMMSlots = 24
MaxPCleSlots = 7
MemoryOperationMode = OptimizerMode
Model = PowerEdge T620
PSRollupStatus = 1
PlatformGUID = 3132334f-c0b7-3480-3510-00364c4c4544
PopulatedCPUSockets = 1
PopulatedDIMMSlots = 1
PopulatedPCleSlots = 1
PowerCap = 336
PowerCapEnabledState = 3
PowerState = 2
PrimaryStatus = 3
RollupStatus = 3
ServerAllocation = null
ServiceTag = 7654321
StorageRollupStatus = 1
SysMemErrorMethodology = 6
SysMemFailOverState = NotInUse
SysMemLocation = 3
SysMemPrimaryStatus = 1
SysMemTotalSize = 2048
SystemGeneration = 12G Monolithic
SystemID = 1231
SystemRevision = 0
TempRollupStatus = 1
UUID = 4c4c4544-0036-3510-8034-b7c04f333231
VoltRollupStatus = 1
smbiosGUID = 44454c4c-3600-1035-8034-b7c04f333231
winrm e "cimv2/root/dcim/DCIM_PSNumericSensor"
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNCheck -SkipCACheck
-encoding:utf-8 -a:basic

OUTPUT:

DCIM_PSNumericSensor
BaseUnits = 6
CreationClassName = DCIM_PSNumericSensor
CurrentReading = 11
CurrentState = Normal
Description = Power Supply Power Consumption
DeviceID = iDRAC.Embedded.1#PS1Current1
ElementName = PS1 Current 1
EnabledDefault = 2
EnabledState = 2
HealthState = 5
LowerThresholdCritical
LowerThresholdNonCritical
OperationalStatus = 2
PossibleStates = Unknown
PossibleStates = Fatal
PossibleStates = Normal
PossibleStates = Upper Fatal
PossibleStates = Upper Critical
PossibleStates = Upper Non-Critical
PossibleStates = Lower Non-Critical
PossibleStates = Lower Critical
PrimaryStatus = 1
RateUnits = 0
RequestedState = 12
Resolution = 1
SensorType = 13
SettableThresholds
SupportedThresholds
SystemCreationClassName = DCIM_ComputerSystem
SystemName = srv:system
TransitioningToState = 12
UnitModifier = -1
UpperThresholdCritical
UpperThresholdNonCritical
ValueFormulation = 2

DCIM_PSNumericSensor
BaseUnits = 7
CreationClassName = DCIM_PSNumericSensor
CurrentReading = 126
CurrentState = Normal
Description = System Power Consumption in Watts
DeviceID = iDRAC.Embedded.1#SystemBoardPwrConsumption
ElementName = System Board Pwr Consumption
EnabledDefault = 2
23 Managing Fiber Channel (FC) Configuration

The Fiber Channel Profile extends the management capabilities of referencing profiles by adding the capability to represent the configuration of fiber channel host bus adapters (FC HBA). The FC HBAs are modeled as views and attributes where there is a view for each individual controller and multiple attributes that allow FC HBA configuration.

Profile and Associated MOFs:
http://www.delltechcenter.com/page/DCIM.Library.Profile

23.1 Listing the FC Inventory-Attribute Class

The FC Inventory contains the following attributes: DCIM_FCIAtribute (23.1), DCIM_FCStatistics(23.2), DCIM_FCString(23.3), DCIM_FCInteger(23.4), and DCIM_FCEnumeration(23.5).

Enumerate FCAtribute class ith the following parameters and syntax:

EXAMPLE:

winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_FCAtribute
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCcheck
-encoding:utf-8 -a:basic

**OUTPUT:**

DCIM_FCString  
AttributeDisplayName  
AttributeName = DeviceName  
CurrentValue = QLogic QLE2662 16Gb FC Adapter  
Dependency  
FQDD = FC.Slot.3-1  
InstanceID = FC.Slot.3-1:DeviceName  
IsReadOnly = true  
MaxLength = 32  
MinLength = 0  
PendingValue  
ValueExpression

DCIM_FCInteger  
AttributeDisplayName  
AttributeName = LinkDownTimeout  
CurrentValue = 30000  
Dependency  
FQDD = FC.Slot.3-2  
InstanceID = FC.Slot.3-2:LinkDownTimeout  
IsReadOnly = false  
LowerBound = 1  
PendingValue  
UpperBound = 255000

**23.2 Listing the FC Inventory-Statistics Class**

If RT-CEM is disabled on the system, this method will return failure.

Enumerate FCStatistics class ith the following parameters and syntax:

**EXAMPLE:**

`winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_FCStatistics  
-u:[USER] -p:[PASSWORD]  
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCcheck  
-encoding:utf-8 -a:basic`

**OUTPUT:**

DCIM_FCStatistics  
FCInvalidCRCs = 0  
FCLinkFailures = 0  
FCLossOfSignals = 0  
FCRxKBCount = 0
FCRxSequences
FCRxTotalFrames = 0
FCTxKBCount = 0
FCTxSequences
FCTxTotalFrames = 0
FQDD = FC.Slot.2-1
InstanceID = FC.Slot.2-1
OSDriverState = 2
PortSpeed = 2
PortStatus = 3

DCIM_FCStatistics
FCInvalidCRCs = 0
FCLinkFailures = 0
FCLossOfSignals = 0
FCRxKBCount = 0
FCRxSequences
FCRxTotalFrames = 0
FCTxKBCount = 0
FCTxSequences
FCTxTotalFrames = 0
FQDD = FC.Slot.2-2
InstanceID = FC.Slot.2-2
OSDriverState = 2
PortSpeed = 2
PortStatus = 3

23.3 Listing the FC Inventory-String Class

Enumerate FCStatistics class with the following parameters and syntax:

**EXAMPLE:**

```shell
winrme http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_FCString
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

DCIM_FCString
AttributeDisplayname
AttributeName = DeviceName
CurrentValue = QLogic QLE2662 16Gb FC Adapter
Dependency
FQDD = FC.Slot.3-1
InstanceID = FC.Slot.3-1:DeviceName
IsReadOnly = true
MaxLength = 32
MinLength = 0
PendingValue
23.4 Listing the FC Inventory-Integer Class

Enumerate FCIInteger class with the following parameters and syntax:

**EXAMPLE:**

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_FCIInteger
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

```
DCIM_FCIInteger
  AttributeDisplayName
  AttributeName = LinkDownTimeout
  CurrentValue = 30000
  Dependency
  FQDD = FC.Slot.3-2
  InstanceID = FC.Slot.3-2:LinkDownTimeout
  IsReadOnly = false
  LowerBound = 1
  PendingValue
  UpperBound = 255000
```

23.5 Listing the FC Inventory-Enumeration Class

Enumerate FCEnumeration class with the following parameters and syntax:

**EXAMPLE:**

```
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_FCEnumeration
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic
```

**OUTPUT:**

```
DCIM_FCEnumeration
  AttributeDisplayName
  AttributeName = PortEnable
  CurrentValue = Disabled
  Dependency
  FQDD = FC.Slot.4-1
  InstanceID = FC.Slot.4-1:PortEnable
  IsReadOnly = false
  PendingValue
```
PossibleValues = Disabled
PossibleValues = Enabled
PossibleValuesDescription

23.6 Changing the FC Attributes-SetAttribute()

The **SetAttribute()** method can be used to change the FC configuration.

Invoke **SetAttribute()** with the following parameters and syntax:

- **TARGET**: Obtained from the *InstanceID* field
- **AttributeName**: Obtained from the *AttributeName* field
- **AttributeValue**: Obtained from the *PossibleValues* field

**EXAMPLE**:  

```
winrm i SetAttribute http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_FCS
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_FCS
+SystemName=DCIM:ComputerSystem
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:SetAttribute_FC.xml
```

The input file *SetAttribute_FC.xml* is shown below:

```
  <p:Target>FC.Slot.2-2</p:Target>
  <p:AttributeName>PortSpeed</p:AttributeName>
  <p:AttributeValue>4G</p:AttributeValue>
</p:SetAttribute_INPUT>
```

**OUTPUT**:

```
SetAttribute_OUTPUT
  Message = The command was successful
  MessageID = FC001
  RebootRequired = Yes
  ReturnValue = 0
  SetResult = Set PendingValue
```

23.7 Applying the Pending Values for FC-CreateTargetedConfigJob()

This method is called to apply the pending values created by the **SetAttribute()** and **SetAttributes()** methods. The system will automatically reboot depending on the *ScheduledStartTime* selected. Using
the `CreateTargetedConfigJob()` `jobID` output with the job control section can be used to obtain its status.

Invoke `CreateTargetedConfigJob()` with the following parameters and syntax:

**TARGET:** This Parameter is the FQDD of the instances, obtained from the `InstanceID` field

**RebootJobType:** There are three options for rebooting the system.

1 = PowerCycle
2 = Graceful Reboot without forced shutdown
3 = Graceful reboot with forced shutdown

Note: When a user does not want to set a reboot type when creating a target job, users should comment out the RebootJobType in the input xml. User should not enter “0” or give no parameter at all in the input xml.

**EXAMPLE:**

```shell
winrm i CreateTargetedConfigJob http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_FCService
?SystemCreationClassName=DCIM_ComputerSystem
+CreationClassName=DCIM_FCService
+SystemName=DCIM:ComputerSystem
+Name=DCIM:FCService
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck
-encoding:utf-8 -a:basic -file:apply_pending_fc.xml
```

The input file `apply_pending_fc.xml` is shown below:

```xml
  <p:Target>FC.Slot.2-2</p:Target>
  <p:RebootJobType>2</p:RebootJobType>
  <p:ScheduledStartTime>TIME_NOW</p:ScheduledStartTime>
  <p:UntilTime>20151111111111</p:UntilTime>
</p:CreateTargetedConfigJob_INPUT>
```

**OUTPUT:**

When this method is executed, a `jobid` or an error message is returned. The status of this `jobid` can be checked within the job control provider in **Section 10**.
23.8 Deleting the Pending Values for FC-DeletePendingConfiguration()
This method is called to cancel the pending values created by the SetAttribute() and SetAttributes() methods. The DeletePendingConfiguration() method cancels the pending configuration changes made before the configuration job is created with CreateTargetedConfigJob(). This method only operates on the pending changes prior to CreateTargetedConfigJob() being called. After the configuration job is created, the pending changes can only be canceled by calling DeleteJobQueue() in the Job Control profile.

Invoke CreateTargetedConfigJob() with the following parameters and syntax:

**Target**: This parameter is the FQDD of the instances

```plaintext
EXAMPLE:
winrm i DeletePendingConfiguration http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_FCService
?SystemCreationClassName=DCIM_ComputerSystem +CreationClassName=DCIM_FCService +SystemName=DCIM:ComputerSystem +Name=DCIM:FCService -u:[USER] -p:[PASSWORD] -r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -file:Delete_Pending_fc.xml
```

The input file `Delete_Pending_fc.xml` is shown below:
```xml
  <p:Target>FC.Slot.2-2</p:Target>
</p:DeletePendingConfiguration_INPUT>
```

**OUTPUT**:

```plaintext
DeletePendingConfiguration_OUTPUT
  Message = The command was successful
  MessageID = FC001
  ReturnValue = 0
  ReturnValue = 4096
```

23.9 Listing the FC Views
Enumerate FCView class ith the following parameters and syntax:

**EXAMPLE**:

```plaintext
winrm e http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_FCView
```
-u:[USER] -p:[PASSWORD]
-r:https://[IPADDRESS]/wsman -SkipCNcheck -SkipCAdcheck
-encoding:utf-8 -a:basic

OUTPUT:

DCIM_FCView
Bus = 5
ChipType = 8324, Rev. 01
Device = 0
DeviceName = QLogic QLE2662 16Gb FC Adapter - 2001000E1E099026
EFIVersion = 5.30
FCTapeEnable = 3
FQDD = FC.Slot.3-1
FabricLoginRetryCount = 0
FabricLoginTimeout = 0
FirstFCTargetLUN = 0
FirstFCTargetWWPN = 00:00:00:00:00:00:00:00
FramePayloadSize = 2048
Function = 0
HardZoneAddress = 0
HardZoneEnable = 3
InstanceID = FC.Slot.3-1
LinkDownTimeout = 30000
LinkStatus = 0
LoopResetDelay = 5
PCIDeviceID = 2031
PortDownRetryCount = 30
PortDownTimeout = 0
PortLoginRetryCount = 8
PortLoginTimeout = 3000
PortNumber = 1
PortSpeed = 2
SecondFCTargetLUN = 0
SecondFCTargetWWPN = 00:00:00:00:00:00:00:00
VendorName
VirtualWWN = 20:00:00:0E:1E:09:90:26
VirtualWWPN = 20:01:00:0E:1E:09:90:26
WWN = 20:00:00:0E:1E:09:90:26
WWPN = 20:01:00:0E:1E:09:90:26