Lifecycle Controller Integration—Best Practices Specification

A Dell Technical White Paper

Steven Zessin
Ganesh Viswanathan
Zhan Liu
Ajay Shenoy
# Lifecycle Controller Integration—Best Practices Specification

## Contents

1. **Contents**
   - How to Use this Document ................................................................. 12
   - Using the Sample Scripts ................................................................. 12
   - Improving winRM Enumeration Performance ............................... 12
   - Feature Discovery Procedure ......................................................... 13
   - Profile Revision Number Explanation ............................................. 14
   - Lifecycle Controller and Lifecycle Controller 2 Nomenclature .......... 15
   - System status and Configuration Job Behavior ............................ 15
     - RS Status and job status ................................................................. 16
     - GetRemoteServicesAPIStatus and job status .............................. 17
     - 11G and 12G compatibility ........................................................... 18
     - GetRemoteServicesAPIStatus output descriptions ....................... 19
   - Reference Links ............................................................................. 20

2. **Terms and Definitions** ................................................................... 21

3. **Anomalous Algorithms and Descriptions** ...................................... 22
   - Setting CNA Bandwidth ................................................................. 22
   - iDRAC Cloning ............................................................................... 22
   - Setting NIC String Attributes to Blank .......................................... 23
   - Determine NIC Card ....................................................................... 23
   - List of Partitionable NIC Cards for LC2 ....................................... 24
   - iDRAC Telnet Attribute Enables SerialRedirection .......................... 24
   - Never Unplug Hardware During Updates ...................................... 25
   - Express Versus Enterprise iDRACs ................................................. 25
   - CIM Query Language (CQL) Filters ............................................. 26
     - CQL filter benefits .................................................................... 26
     - How to perform CQL enumerations using RECITE ..................... 26
     - CQL filter example that enumerates all NIC attributes for a particular port/partition 26
     - CQL filter example that enumerates all iDRAC attributes with a particular GroupDisplayName and AttributeName ........................................ 27
     - CQL filter example that enumerates all BIOS attributes where the IsReadOnly flag is set to TRUE 27
   - Ordering of iDRAC Attributes to set or apply (11Generation vs 12Generation) 27

3.10 How to Determine if Server is 11Generation vs 12Generation .......... 28
3.11 Using Special Characters in Usernames ......................................... 28
3.12 Obtaining Updated System Inventory ........................................... 28
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.14</td>
<td>How to Determine if the System is Blade, Tower, or Rack</td>
<td>30</td>
</tr>
<tr>
<td>3.15</td>
<td>Getting the SystemID, Model, and more</td>
<td>30</td>
</tr>
<tr>
<td>3.16</td>
<td>Http, CIFS, NFS, tftp, ftp Formatting</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>Workflows</td>
<td>32</td>
</tr>
<tr>
<td>4.1</td>
<td>RAID stacking: ResetConfig, CreateVD, assign HotSpares</td>
<td>32</td>
</tr>
<tr>
<td>4.2</td>
<td>RAID Stacking with BIOS Attributes Using Setupjobqueue</td>
<td>33</td>
</tr>
<tr>
<td>4.3</td>
<td>Boot to Network ISO</td>
<td>35</td>
</tr>
<tr>
<td>4.4</td>
<td>Boot to ISO from vFlash</td>
<td>36</td>
</tr>
<tr>
<td>4.5</td>
<td>Set Hard Disk Drive to ‘first’ in Boot Order</td>
<td>37</td>
</tr>
<tr>
<td>4.6</td>
<td>Export (backup) Image to vFlash</td>
<td>38</td>
</tr>
<tr>
<td>4.7</td>
<td>Export (backup) ilmage to CIFS or NFS Share</td>
<td>38</td>
</tr>
<tr>
<td>4.8</td>
<td>Automatic Backup (12th Generation and Later Version of Servers Only)</td>
<td>39</td>
</tr>
<tr>
<td>4.9</td>
<td>Import (restore) Image from vFlash</td>
<td>40</td>
</tr>
<tr>
<td>4.10</td>
<td>Import (Restore) Image from CIFS or NFS Share</td>
<td>40</td>
</tr>
<tr>
<td>4.11</td>
<td>iDRAC Firmware DUP uUpdate from CIFS or TFTP Share</td>
<td>41</td>
</tr>
<tr>
<td>4.12</td>
<td>BIOS Firmware DUP Update from CIFS or TFTP Share</td>
<td>42</td>
</tr>
<tr>
<td>4.13</td>
<td>USC Firmware DUP Update from CIFS or TFTP Share</td>
<td>43</td>
</tr>
<tr>
<td>4.14</td>
<td>Automatic Firmware Update (12th Generation and Later Version of Servers Only)</td>
<td>43</td>
</tr>
<tr>
<td>4.15</td>
<td>Update from Repository (12th Generation and Later Version of Servers Only)</td>
<td>44</td>
</tr>
<tr>
<td>4.16</td>
<td>Firmware Rollback (12th Generation and Later Version of Servers Only)</td>
<td>45</td>
</tr>
<tr>
<td>4.17</td>
<td>Remote Diagnostics (12th Generation and Later Version of Servers Only)</td>
<td>45</td>
</tr>
<tr>
<td>4.18</td>
<td>PXE Boot using Embedded NICs (11G only)</td>
<td>46</td>
</tr>
<tr>
<td>4.19</td>
<td>PXE Boot using Embedded NICs (12G only)</td>
<td>48</td>
</tr>
<tr>
<td>4.20</td>
<td>Set NIC Attributes and iSCSI boot using setupjobqueue (11G only)</td>
<td>49</td>
</tr>
<tr>
<td>4.21</td>
<td>iSCSI Boot using NDC/Broadcom (12G only)</td>
<td>51</td>
</tr>
<tr>
<td>4.22</td>
<td>iSCSI Boot using QLogic (12G only)</td>
<td>52</td>
</tr>
<tr>
<td>4.23</td>
<td>iSCSI boot using Intel (12th Generation only)</td>
<td>54</td>
</tr>
<tr>
<td>4.24</td>
<td>IO Identity</td>
<td>56</td>
</tr>
<tr>
<td>4.25</td>
<td>Export LC log</td>
<td>57</td>
</tr>
<tr>
<td>4.26</td>
<td>FCoE Boot using QLogic (12G only)</td>
<td>58</td>
</tr>
<tr>
<td>4.27</td>
<td>FCoE boot using Intel (12th Generation only)</td>
<td>60</td>
</tr>
<tr>
<td>4.28</td>
<td>FCoE boot using Broadcom (12G only)</td>
<td>63</td>
</tr>
<tr>
<td>4.29</td>
<td>IO Identity for QLogic (12G only)</td>
<td>64</td>
</tr>
<tr>
<td>4.30</td>
<td>IO Identity for Broadcom (12G only)</td>
<td>66</td>
</tr>
<tr>
<td>4.31</td>
<td>IO Identity for Intel (12G only)</td>
<td>68</td>
</tr>
</tbody>
</table>
4.32 Export System Configuration (12th Generation and Later Version of Servers Only) ......... 69
4.33 Import System Configuration (12th Generation and Later Version of Servers Only) ........ 69
4.34 Configurable Boot to Network ISO ................................................................. 70
5 Base Metrics Profile Use Cases .............................................................................. 71
5.1 Discovery of Base Metrics Profile Support ....................................................... 71
6 BIOS and Boot Management Profile Use Cases ..................................................... 72
6.1 Discovery of BIOS and boot Profile Support ..................................................... 72
6.2 List all BIOS Attributes .................................................................................. 73
6.3 Delete Pending BIOS Configuration ............................................................... 73
6.4 Inventory of boot Configurations in System .................................................... 74
6.5 Get the First boot Configuration’s Information ................................................ 74
6.6 Inventory of boot Sources in System ............................................................... 75
6.7 Changing boot Order by Instance ................................................................. 75
6.8 Enable or Disable boot Source ................................................................. 75
6.9 One Time boot ...................................................................................... 76
7 CPU Profile Use Cases ............................................................................................ 77
7.1 Discovery of CPU Profile Support .................................................................... 77
7.2 Inventory of CPUs in System ......................................................................... 78
7.3 Get the First CPU’s Information ...................................................................... 78
8 Event Filter Profile Use Cases .................................................................................. 78
8.1 Discovery of Event Filter Profile Support ........................................................ 78
8.2 Get Event Filter Configuration Service Views ................................................ 79
8.3 Get Event Filter Views .................................................................................. 80
8.4 Get Single Event Filter’s Information ............................................................. 80
8.5 Set Event Filters by Category ......................................................................... 80
8.6 Set Event Filters by InstanceID ....................................................................... 81
9 iDRAC Card Profile Use Cases ................................................................................ 81
9.1 Discovery of iDRAC Card Profile Support ....................................................... 81
9.2 Get all iDRAC Card Attributes ...................................................................... 82
9.3 Inventory of iDRAC Cards in System ............................................................ 82
9.4 Get the First iDRAC Card’s Information ......................................................... 83
9.5 Set Apply iDRAC Card Attribute(s) Immediately ............................................. 83
9.6 Schedule a set iDRAC Card Attribute(s) Operation ....................................... 84
10 Fan Profile Use Cases ....................................................................................... 84
10.1 Discovery of Fan Profile Support ................................................................. 84
10.2 Inventory of Fans in System ................................................................. 85
10.3 Get the First Fan’s Information .............................................................. 85
11 Persistent Storage Profile Use Cases ........................................................... 86
11.1 Discovery of Persistent Storage Profile Support ......................................... 86
11.2 Inventory of Virtual Flash (vFlash) Media .................................................. 86
11.3 Get the First vFlash’s Attribute Information ............................................... 87
11.4 Inventory of Partitions on the Virtual Flash Media ....................................... 87
11.5 Initialize Virtual Flash Media ................................................................. 88
11.6 Enable Virtual Flash (vFlash) Media ........................................................ 88
11.7 Disable Virtual Flash (vFlash) Media ........................................................ 88
11.8 Create new Partition on Virtual Flash (vFlash) Media ............................... 89
11.9 Create new Partition Using Image ........................................................... 89
11.10 Delete Existing Partition ....................................................................... 90
11.11 Format Existing Partition ....................................................................... 90
11.12 Modify Existing Partition ....................................................................... 91
11.13 Attach Partition ..................................................................................... 91
11.14 Detach Partition .................................................................................... 92
11.15 Export Data from Existing Partition ......................................................... 92
12 Power State Management Profile Use Cases ................................................. 93
12.1 Discovery of Power State Management Profile Support .............................. 93
13 Profile Registration Profile Use Cases .......................................................... 94
13.1 Discovery of Profile Registration Profile Support ....................................... 94
14 Simple RAID Profile Use Cases ................................................................. 94
14.1 Discovery of RAID Profile Support .......................................................... 94
14.2 Inventory of RAID Controllers in System .................................................. 95
14.3 Get the first RAID Controller’s Information .............................................. 96
14.4 Inventory of Virtual and Physical Disk Drives in System ............................ 96
14.5 Apply Pending Values for a RAID Configuration ....................................... 96
14.6 Delete Pending Values for a RAID Configuration ....................................... 96
14.7 Clear old Configuration from Newly Added HDD ...................................... 97
14.8 Determine Available RAID Configurations .............................................. 97
14.9 Determine Available Physical Disk Drives for a RAID Configuration .......... 98
14.10 Check Available Virtual Disk Parameters for a given RAID Level and set of Physical Disks . 98
14.11 Create a Virtual Disk ............................................................................ 99
14.12 Determine Available Physical Disk Drives to be used as a Hot-spare.......... 99
### Lifecycle Controller Integration—Best Practices Specification

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.13</td>
<td>Assign a Physical Disk Drive as a Hot-spare</td>
</tr>
<tr>
<td>14.14</td>
<td>Delete a Virtual Disk from the System</td>
</tr>
<tr>
<td>14.15</td>
<td>Delete all Virtual Disks and Unassign all Hot-spares</td>
</tr>
<tr>
<td>14.16</td>
<td>Convert Physical Disk Drive to RAID State</td>
</tr>
<tr>
<td>14.17</td>
<td>Convert Physical Disk Drives to non-RAID State</td>
</tr>
<tr>
<td>15.1</td>
<td>Discovery of Record Log Profile Support</td>
</tr>
<tr>
<td>15.2</td>
<td>List Lifecycle Record Logs</td>
</tr>
<tr>
<td>15.3</td>
<td>List Lifecycle Record Log Capabilities</td>
</tr>
<tr>
<td>15.4</td>
<td>List Lifecycle Log Entries</td>
</tr>
<tr>
<td>15.5</td>
<td>Set and get Comment in Lifecycle Log Entries</td>
</tr>
<tr>
<td>15.6</td>
<td>List System Event Record Logs</td>
</tr>
<tr>
<td>15.7</td>
<td>List System Event Record Log Capabilities</td>
</tr>
<tr>
<td>15.8</td>
<td>List System Event Log Entries</td>
</tr>
<tr>
<td>16.1</td>
<td>Discovery of RBAP Profile Support</td>
</tr>
<tr>
<td>16.2</td>
<td>Discovery of Users with Assigned LAN Privileges</td>
</tr>
<tr>
<td>16.3</td>
<td>Discovery of Users with Assigned Serial over LAN Privileges</td>
</tr>
<tr>
<td>16.4</td>
<td>Discovery of users with Assigned CLP Privileges</td>
</tr>
<tr>
<td>17.1</td>
<td>Discovery of Service Processor Profile Support</td>
</tr>
<tr>
<td>18.1</td>
<td>Discovery of Simple NIC Profile Support</td>
</tr>
<tr>
<td>18.2</td>
<td>Inventory of NICs in System</td>
</tr>
<tr>
<td>18.3</td>
<td>Get the First NIC’s Information</td>
</tr>
<tr>
<td>18.4</td>
<td>List all NIC Attributes</td>
</tr>
<tr>
<td>18.5</td>
<td>Delete Pending NIC Values</td>
</tr>
<tr>
<td>18.6</td>
<td>Discovery of NIC Capabilities</td>
</tr>
<tr>
<td>19.1</td>
<td>Discovery of Software Update Profile Support</td>
</tr>
<tr>
<td>20.1</td>
<td>Discovery of Job Control Profile Support</td>
</tr>
<tr>
<td>20.2</td>
<td>List all Jobs in Job Store</td>
</tr>
<tr>
<td>20.3</td>
<td>Get a Job’s Information</td>
</tr>
<tr>
<td>20.4</td>
<td>Delete all Jobs from Job Store (job queue) using “JID_CLEARALL”</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>28.8</td>
<td>Delete Pending System Attribute Values</td>
</tr>
<tr>
<td>29</td>
<td>Software Inventory Profile Use Cases</td>
</tr>
<tr>
<td>29.1</td>
<td>Instance Diagram</td>
</tr>
<tr>
<td>29.2</td>
<td>Discovery of Software Inventory Profile Support</td>
</tr>
<tr>
<td>29.3</td>
<td>Inventory of Software in System</td>
</tr>
<tr>
<td>29.4</td>
<td>Get the Installed BIOS Firmware Inventory</td>
</tr>
<tr>
<td>29.5</td>
<td>Get the Available iDRAC Firmware Inventory</td>
</tr>
<tr>
<td>30</td>
<td>Simple Identity Management Profile Use Cases</td>
</tr>
<tr>
<td>30.1</td>
<td>Discovery of Simple Identity Management Profile Support</td>
</tr>
<tr>
<td>31</td>
<td>LC Management Profile Use Cases</td>
</tr>
<tr>
<td>31.1</td>
<td>Discovery of LC Management Profile Support</td>
</tr>
<tr>
<td>31.2</td>
<td>Inventory of LC Management Attributes in system</td>
</tr>
<tr>
<td>31.3</td>
<td>Check and enable (or disable) Collect System Inventory on Restart (CSIOR)</td>
</tr>
<tr>
<td>31.4</td>
<td>Check Version of Lifecycle Controller (LC)</td>
</tr>
<tr>
<td>31.5</td>
<td>Get “Part Firmware Update” Attribute</td>
</tr>
<tr>
<td>31.6</td>
<td>Check vFlash License Enablement</td>
</tr>
<tr>
<td>31.7</td>
<td>Set Configuration to “Auto Discovery Factory Defaults”</td>
</tr>
<tr>
<td>31.8</td>
<td>Clear Provisioning Server</td>
</tr>
<tr>
<td>31.9</td>
<td>Replace Auto Discovery Public Key</td>
</tr>
<tr>
<td>31.10</td>
<td>Replace auto Discovery Client Certificate, Private key and Password</td>
</tr>
<tr>
<td>31.11</td>
<td>Delete auto Discovery Public Key</td>
</tr>
<tr>
<td>31.12</td>
<td>Delete auto Discovery Client Certificate, Private Key and Password</td>
</tr>
<tr>
<td>31.13</td>
<td>Replace iDRAC Web Server Client Certificate and Private Key</td>
</tr>
<tr>
<td>31.14</td>
<td>Replace iDRAC Web Server Public Certificate</td>
</tr>
<tr>
<td>31.15</td>
<td>Insert Comment into Lifecycle Log</td>
</tr>
<tr>
<td>31.16</td>
<td>Export and View the Content of the Lifecycle Log</td>
</tr>
<tr>
<td>31.17</td>
<td>Export and View the Current Hardware Inventory</td>
</tr>
<tr>
<td>31.18</td>
<td>Export and View the Hardware Inventory as Shipped from the Factory</td>
</tr>
<tr>
<td>32</td>
<td>OS Deployment Profile Use Cases</td>
</tr>
<tr>
<td>32.1</td>
<td>Discovery of OS Deployment Profile Support</td>
</tr>
<tr>
<td>32.2</td>
<td>Unpack and Attach Drivers</td>
</tr>
<tr>
<td>32.3</td>
<td>Connect and Attach Network ISO Image</td>
</tr>
<tr>
<td>32.4</td>
<td>Disconnect and Detach Network ISO Image</td>
</tr>
<tr>
<td>32.5</td>
<td>Get ISO Image Connection Status</td>
</tr>
<tr>
<td>32.6</td>
<td>One-time ISO boot Skip</td>
</tr>
</tbody>
</table>
32.7 Remote File Share (RFS) Use Cases ................................................................. 145

32.7.1 Connect and Attach Network ISO Image as a USB CD-ROM device via RFS USB end point. 145
32.7.2 Disconnect and detach ISO Image exposed via RFS USB end point .................... 145
32.7.3 Get RF ISO Image connection Status ........................................................... 145

32.8 Boot to Hard Drive (HD) .............................................................................. 146

33 Appendix ................................................................................................ 147

33.1 PYTHON scripts README .............................................................................. 147

33.1.1 Purpose.............................................................................................. 147
33.1.2 Requirements ...................................................................................... 147
33.1.3 Command line ...................................................................................... 147
33.1.4 Commands .......................................................................................... 148
33.1.5 Settable variables ................................................................................. 150
33.1.6 Internal variables .................................................................................. 151

33.2 System check information ............................................................................ 152

33.2.1 Check System Power State ....................................................................... 152
33.2.2 Check RS status .................................................................................... 152
33.2.3 Check for pending jobs ........................................................................... 152
33.2.4 Check for pending configuration ............................................................ 152
33.2.5 Check CSIOR state ................................................................................. 152

33.3 Inventory information ................................................................................. 152

33.3.1 System inventory .................................................................................. 152
33.3.2 Software inventory ................................................................................. 153
33.3.3 BIOS inventory ..................................................................................... 153
33.3.4 Boot order inventory .............................................................................. 153
33.3.5 NIC inventory ....................................................................................... 153
33.3.6 RAID inventory ..................................................................................... 153

33.4 Poll LC jobs information .............................................................................. 153

33.4.1 Timing considerations ............................................................................ 153
33.4.2 Machine reboot ..................................................................................... 153
33.4.3 POST ................................................................................................... 154
33.4.4 SSM ................................................................................................... 154
33.4.5 RSStatus/JobStatus .............................................................................. 155
33.4.6 Check refreshed data ............................................................................. 155
33.4.7 CSIOR ................................................................................................. 155
Lifecycle Controller Integration—Best Practices Specification

33.5 iSCSI boot information

Tables
Table 1. Generational Nomenclature

Figures
Figure 1. Typical Life Cycle of an 11G (LC1.5.0 & LC1.5.1) Configuration Job
Figure 2. Typical Life Cycle of a 12G Configuration Job
Figure 3. Compatibility with 11G Workflows
Figure 4. Software Inventory: Instance Diagram
1 Contents

1.1 How to Use this Document
This document contains the detailed steps of common workflows to perform various tasks utilizing
winRM or WSMAN. The PYTHON scripting language was used to provide a software development kit
(SDK) for Lifecycle Controller (LC) API methods. Two primary objectives are addressed: first, that
workflows documentation provides guidance to established, known, working API methodologies, and
second, that corresponding PYTHON sample scripts are separately provided. After invoking these
scripts, the output log can be used to provide approximate timing on a particular system configuration,
as well as raw winRM or WSMAN input and output.

1.2 Using the Sample Scripts
Refer to the appendix for a full text README.

Getting started:
1) Install Python 2.4 to 2.6
2) Download scripts folder to desired location; no installation necessary

Running a script:
1) cd to scripts directory
2) python recite.py (opens command prompt of application)
3) set $IP 12.34.56 (Enter actual IP)
4) log whateverfilename.log w (may use any name for filename)
5) batch bestpracticeflows\script_name.win (launches script)

Note: A .win file is simply a text file containing calls to the recite PYTHON script.

Other commands:
-help (lists all available commands)
-set (list current IP, username, password, etc.)

Notes:
File output will be placed in scripts directory

1.3 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.

Also see section 3.9 for using CQL filters on enumerations.

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

winrm set winrm/config @{MaxBatchItems=“50”}
Additionally, increasing the allotted maximum envelope size and timeout can also increase performance.

```
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.

```
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:

```
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*.

```
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.

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

### 1.4 Feature Discovery Procedure

There are four steps recommended to determining the feature set on a given system.

1. Interop namespace - registered profile advertisement
   b. The output from above will provide the *RegisteredVersion* of each supported profile on the system. The *RegisteredVersion* field can be used to determine the profile’s feature set. See Section 1.4 for more information.

2. Capability properties on views (NIC example)
   b. The output from above will provide the available properties of each view.

3. Capability Attributes (RAID example)
a. Enumerate the DCIM_RAIDString, DCIM_RAIDEnumeration, and DCIM_RAIDInteger classes (DCIM_RAIDString shown below)

\[
\text{winrm enumerate "cimv2/root/dcim/DCIM_RAIDString" -r:https://IPADDRESS/wsman=-u:username -p:password -SkipCNcheck -SkipCAcheck -encoding:(utf)-8 -a:basic -format:pretty}
\]

b. The output from above will provide the available attributes of each class

4) Firmware versioning


b. The output from above will list all available and installed firmwares and corresponding firmware versions. Examining the version of these firmwares, such as Lifecycle controller and iDRAC, can be used to determine the feature set

### 1.5 Profile Revision Number Explanation

Profile revision numbers are a key metric in determining the available feature set. Examples of discovering, or obtaining, profiles are covered extensively in this document along with example output.

The example shown below is for the LC Management profile. Discovering a particular profile on a system is a three step process.

1) Enumerate the DCIM_LCRegisteredProfile class to view all available profiles
2) Search the RegisteredName field for the desired profile
3) Search for the RegisteredVersion field, which is the characteristic used to identify the supported features.

**DCIM_LCRegisteredProfile**
- AdvertiseTypeDescriptions = WS-Identify
- AdvertiseTypeDescriptions = Interop Namespace
- AdvertiseTypes = 1
- AdvertiseTypes = 1
- InstanceID = DCIM:LCManagement:1.1.0
- OtherRegisteredOrganization = DCIM
- ProfileRequireLicense = Auto Discovery
- ProfileRequireLicense = Part Replacement
- ProfileRequireLicense = Remote Firmware Configuration
- ProfileRequireLicense = Remote Inventory Export
- ProfileRequireLicense = Server Profile Export and Import
- ProfileRequireLicenseStatus = LICENSED
- ProfileRequireLicenseStatus = LICENSED
- ProfileRequireLicenseStatus = LICENSED
- ProfileRequireLicenseStatus = LICENSED
- ProfileRequireLicenseStatus = LICENSED
- RegisteredName = LC Management
- RegisteredOrganization = 1
- RegisteredVersion = 1.4.0

The RegisteredVersion field is in the following format:
[major change] . [minor change] . [errata]

- Increments in the major change field indicate that the profile is not backward compatible.
- Increments in the minor change field indicate that one or more new methods have been added.
- Increments in the errata field indicate that one more defects have been fixed.

### 1.6 Lifecycle Controller and Lifecycle Controller 2 Nomenclature

This section describes the new terminology associated with the new generation of hardware and Lifecycle Controller firmware (including iDRAC). The new hardware platform is generically referred to as 12G and all corresponding Lifecycle Controller firmware will be LC2 with accompanying sub releases (i.e. 1.0.0, 1.1.0, etc.). The table below summarizes both the past and current generational nomenclature.

**NOTE**: Data within table is for illustration purposes only.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Lifecycle Controller Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>11G</td>
<td>LC 1.5.0</td>
</tr>
<tr>
<td>11G</td>
<td>LC 1.5.1</td>
</tr>
<tr>
<td>11G</td>
<td>LC 1.5.2</td>
</tr>
<tr>
<td>11G</td>
<td>LC 1.x.x</td>
</tr>
<tr>
<td>12G</td>
<td>LC2 1.0.0</td>
</tr>
<tr>
<td>12G</td>
<td>LC2 1.0.1</td>
</tr>
<tr>
<td>12G</td>
<td>LC2 1.1.0</td>
</tr>
<tr>
<td>12G</td>
<td>LC2 1.1.1</td>
</tr>
<tr>
<td>12G</td>
<td>LC2 1.x.x</td>
</tr>
</tbody>
</table>

Changes in the Lifecycle Controller firmware versions abide by the following definition:

[major change] . [minor change] . [errata]

- Increments in the major change field indicate that the profile is not backward compatible.
- Increments in the minor change field indicate that one or more new methods have been added.
- Increments in the errata field indicate that one more defects have been fixed.

### 1.7 System status and Configuration Job Behavior

The details below describe the generational evolution of how a typical configuration job relates to the state of the system.
1.7.1 **RS Status and job status**

The details below describe how the remote service (RS) status relates to the job status. RS status is a feature that indicates whether the system is ready to invoke WSMAN commands. It must be in a *ready* state before executing any WSMAN commands.

**NOTE:** The RS Status method was introduced in LC1.5.0.

**Figure 1. Typical Life Cycle of an 11G (LC1.5.0 & LC1.5.1) Configuration Job**
After the job is complete in Automated Task Application (previously SSM), the job status is immediately updated in the job store. The job is moved immediately to the **Completed** state once it is complete in the Automated Task Application. As seen in the timeline diagram above, after the job is **Completed**, the sync happens in the configDB, and then the RS status goes to **Reloading** state. After all the required populators are refreshed successfully, the RS status goes to **Ready** state. The user/console can see the new values only when the RS status goes to the **Ready** state.

### 1.7.2 GetRemoteServicesAPIStatus and job status

The introduction of the new GetRemoteServicesAPIStatus method alleviates the ambiguity of the GetRSStatus method regarding when the system is **ready**.

**Figure 2. Typical Life Cycle of a 12G Configuration Job**

<table>
<thead>
<tr>
<th>Job Status</th>
<th>GetRSStatus()</th>
<th>GetRemoteServicesAPIStatus()</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ready</strong></td>
<td><strong>Sync</strong></td>
<td><strong>Lifestyle Controller</strong></td>
</tr>
<tr>
<td><strong>Running</strong></td>
<td><strong>Reloading</strong></td>
<td><strong>Remote Services is not ready.</strong></td>
</tr>
<tr>
<td><strong>Completed</strong></td>
<td><strong>READY</strong></td>
<td><strong>Lifestyle Controller</strong></td>
</tr>
<tr>
<td><strong>Completed</strong></td>
<td><strong>RELOADING</strong></td>
<td><strong>Remote Services is ready.</strong></td>
</tr>
</tbody>
</table>

**Note:** Dotted lines denote the old behavior. Bold red lines denote the new behavior in 12G.

After the job is complete in the Automated Task Application, those jobs that require a refresh - jobs that have message IDs - JOB_SUCCESS and JOB_COMPLETED_ERROR will be kept in **RUNNING state** till the new sync comes in and the Data Manager is moved to **READY state**.

Jobs that don’t require a refresh will be moved to complete immediately once the job is complete in the Automated Task Application.
1.7.3 11G and 12G compatibility

Figure 3. Compatibility with 11G Workflows

NOTE: Dotted lines denote the old behavior. Bold red line denotes the new behavior in 12G.

The existing 11G workflows expect the RS status to be in RELOADING state once the job is marked COMPLETED. So to maintain the compatibility with the 11G workflows, the RS status is artificially held in the RELOADING state for 90 seconds even though it is actually READY. This time limit was provided by the console team.
### 1.7.4 GetRemoteServicesAPIStatus output descriptions

<table>
<thead>
<tr>
<th>Output parameter Name</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status</strong></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><strong>MessageID</strong></td>
<td>LC060</td>
<td>Message for ID LC060</td>
</tr>
<tr>
<td></td>
<td>LC061</td>
<td>Message for ID LC061</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>Lifecycle Controller Remote Services is not ready. Message for ID LC060</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lifecycle Controller Remote Services is ready. Message for ID LC061</td>
<td></td>
</tr>
<tr>
<td><strong>ServerStatus</strong></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></td>
<td>5 (Lifecycle Controller Unified Server Configurator)</td>
<td>Server is executing UEFI Lifecycle Controller Unified Server Configurator application</td>
</tr>
<tr>
<td><strong>LCStatus</strong></td>
<td>0 (Ready)</td>
<td>Lifecycle Controller instrumentation is up to date and enabled</td>
</tr>
<tr>
<td></td>
<td>1 (Not Initialized)</td>
<td>Lifecycle Controller instrumentation is not initialized. The initialization operation may take up to a minute.</td>
</tr>
<tr>
<td></td>
<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>
</tr>
<tr>
<td></td>
<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>
</tr>
<tr>
<td></td>
<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>
</tr>
<tr>
<td></td>
<td>5 (In Use)</td>
<td>Lifecycle Controller is being currently used by another process.</td>
</tr>
</tbody>
</table>
1.8 Reference Links

Web Services Interface Guide for Windows & linux
- [http://www.delltechcenter.com/page/Lifecycle+Controller](http://www.delltechcenter.com/page/Lifecycle+Controller)

Profiles

PCI ID reference list
- [http://pciids.sourceforge.net/pci.ids](http://pciids.sourceforge.net/pci.ids)

Installation and Configuration for Windows Remote Management
# Terms and Definitions

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2.1</strong></td>
<td><strong>LC</strong></td>
</tr>
<tr>
<td></td>
<td>Lifecycle Controller</td>
</tr>
<tr>
<td><strong>2.2</strong></td>
<td><strong>ENUMERATE</strong></td>
</tr>
<tr>
<td></td>
<td>Refers to WS-MAN <strong>ENUMERATE</strong> operation as described in Section 8.2 of DSP0226_V1.1 and Section 9.1 of DSP0227_V1.0</td>
</tr>
<tr>
<td><strong>2.3</strong></td>
<td><strong>GET</strong></td>
</tr>
<tr>
<td></td>
<td>Refers to WS-MAN <strong>GET</strong> operation as defined in Section 7.3 of DSP0226_V1.1 and Section 7.1 of DSP0227_V1.0</td>
</tr>
<tr>
<td><strong>2.4</strong></td>
<td><strong>iDRAC</strong></td>
</tr>
<tr>
<td></td>
<td>integrated Dell Remote Access Controller – management controller for blades and monolithic servers</td>
</tr>
<tr>
<td><strong>2.5</strong></td>
<td><strong>USC</strong></td>
</tr>
<tr>
<td></td>
<td>Unified Server Configurator</td>
</tr>
<tr>
<td><strong>2.6</strong></td>
<td><strong>iSCSi</strong></td>
</tr>
<tr>
<td></td>
<td>Internet Small Computer System Interface, an Internet Protocol (IP)-based storage networking standard for linking data storage facilities.</td>
</tr>
<tr>
<td><strong>2.7</strong></td>
<td><strong>SSM</strong></td>
</tr>
<tr>
<td></td>
<td>System Services Manager</td>
</tr>
<tr>
<td><strong>2.8</strong></td>
<td><strong>CSIOR</strong></td>
</tr>
<tr>
<td></td>
<td>Collect System Inventory on Restart</td>
</tr>
<tr>
<td><strong>2.9</strong></td>
<td><strong>SSIB</strong></td>
</tr>
<tr>
<td></td>
<td>System Services Information Block</td>
</tr>
<tr>
<td><strong>2.10</strong></td>
<td><strong>UEFI</strong></td>
</tr>
<tr>
<td></td>
<td>Unified Extensible Firmware Interface</td>
</tr>
<tr>
<td><strong>2.11</strong></td>
<td><strong>BIOS</strong></td>
</tr>
<tr>
<td></td>
<td>Basic Input / Output System</td>
</tr>
<tr>
<td><strong>2.12</strong></td>
<td><strong>NIC</strong></td>
</tr>
<tr>
<td></td>
<td>Network Interface Controller</td>
</tr>
<tr>
<td><strong>2.13</strong></td>
<td><strong>FQDD</strong></td>
</tr>
<tr>
<td></td>
<td>Fully Qualified Device Description</td>
</tr>
<tr>
<td><strong>2.14</strong></td>
<td><strong>LCL</strong></td>
</tr>
<tr>
<td></td>
<td>Lifecycle Log</td>
</tr>
<tr>
<td><strong>2.15</strong></td>
<td><strong>WSIG</strong></td>
</tr>
<tr>
<td></td>
<td>Web Services Interface Guide</td>
</tr>
</tbody>
</table>
3 Anomalous Algorithms and Descriptions

3.1 Setting CNA Bandwidth
The recommended algorithm is to schedule jobs such that MinBandwidths are first reduced, and then increased. Essentially, delta values need to be sorted, and jobs scheduled in that order.

Consider the following example:

Current: 25, 25, 25, 25
Target: 30, 30, 20, 20

Since 20, 20 are both reducing values from 25, they should be scheduled first. This makes space for increasing the other values. Next, values being increased can be scheduled - 30, 30.

A more complex example:

FQDD: 1, 2, 3, 4
Current: 5, 5, 50, 40
Target: 40, 50, 5, 5
Deltas: -35, -45, 45, 35
Order of job FQDDs: 3, 4, 1, 2

MinBandwidth limitations are documented in the Simple NIC Profile in section 6.7.

http://attachments.wetpaintserv.us/chcAMan6fnC1y828qYSnag350700

3.2 iDRAC Cloning
Blade cloning consists of a pull, enumerating attributes, and a push, applying attributes. The pull command is a basic enumeration command using the DCIM_iDRACCardAttribute class.

Applies to: LC2+

Script: iDRACClonePull.win

A) The remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) ENUMERATE the DCIM_iDRACCardAttribute class (same as DCIM_iDRACCardEnumeration class) and store the results to be pushed. See section 2.2 for a definition of ENUMERATE.

The blade cloning push requires an extended WSMAN timeout of 120 seconds, versus the default of 60 seconds. This is needed because of the numerous amounts of attributes that need to be applied.

Below are Windows (winRM) and Linux (wsman) examples that apply all the iDRAC attributes. Replace [IP_ADDRESS], [USER_NAME], and [PASS_WORD] with the actual IP address, username, and password.

winRM format:
Lifecycle Controller Integration—Best Practices Specification


WSMAN format:


Partial Example of iDRAC.Embedded.1_setatts.xml

```xml
  <p:Target>iDRAC.Embedded.1</p:Target>
  <p:AttributeName>Info.1#Product</p:AttributeName>
  <p:AttributeValue>Integrated Dell Remote Access Controller</p:AttributeValue>
  <p:AttributeName>Info.1#Version</p:AttributeName>
  <p:AttributeValue>1.0.0</p:AttributeValue>
</p:ApplyAttributes_INPUT>
```

3.3 Setting NIC String Attributes to Blank

An issue exists when setting NIC String parameters, such as `IscsiInitiatorName`, using the NIC menu (usually entered via ctrl-s) from a non-blank value to a blank value.

As a result, when the NIC is used for operations that utilize this parameter, such as iscsi boot, the operation will be unsuccessful because it will use the old value instead of the one displayed through WSMAN.

The work around for this scenario is to not set any NIC String parameters to a blank value.

3.4 Determine NIC Card

There are two different ways to determine the model/type of a NIC card:
1) **ENUMERATE** the *DCIM_NICView* class and note the PCID, then look up in PCID table [http://pciids.sourceforge.net/pci.ids](http://pciids.sourceforge.net/pci.ids)

2) **ENUMERATE** the *DCIM_NICString* class and search for attribute *ChipMdl*. The *CurrentValue* parameter will contain the NIC card model number.

```
DCIM_NICString

AttributeName = ChipMdl
CurrentValue = BCM5716 C0
DefaultValue
FQDD = NIC.Embedded.2-1
InstanceID = NIC.Embedded.2-1:ChipMdl
IsReadOnly = true
MaxLength = 0
MinLength = 0
PendingValue
```

### 3.5 List of Partitionable NIC Cards for LC2

The following is a list of partitionable NICs supported by LC2.

**Broadcom**

57810 bNDC - JVFR
57810 SFP+ NIC - N20KJ
57810 Base-T NIC - W1GCR
57810 Mezz - 55GHP
57800 rNDC - MT09V

**QLogic**

QMD8262 bNDC - D90TX
QLE8262 NIC - JHD51
QME8262 Mezz - 9Y65N

### 3.6 iDRAC Telnet Attribute Enables SerialRedirection

The following behavior describes a situation where setting certain iDRAC attribute(s) causes another iDRAC attribute to automatically and simultaneously change.

Setting any of the following *Telnet* attributes causes the *SerialRedirection* attribute to become *Enabled*.

- Port  [ InstanceID = iDRAC.Embedded.1#Telnet.1#Port ]
- Enable  [ InstanceID = iDRAC.Embedded.1#Telnet.1#Enable ]
- Timeout  [ InstanceID = iDRAC.Embedded.1#Telnet.1#Timeout ]
Recommendation:
Change the Telnet attributes prior to setting the `SerialRedirection.Enable` attribute. Or, if using input XML, have the Telnet attributes before the `SerialRedirection.Enable` attribute.

3.7 Never Unplug Hardware During Updates

Users should not unplug any hardware during critical remote enablement (RE) updates. This may result in unexpected behaviors.

As an example, unplugging a USB key during critical updates may cause a Red Screen of Death (RSOD).

3.8 Express Versus Enterprise iDRACs

There are four levels of iDRAC licensing as follows:

1) Basic
2) Express (Monolithic)
3) Express for blades (Modular)
4) Enterprise

One method of determining the level on a system is to perform an enumeration of the `DCIM_iDRACCardView` class. Example results are shown for reference.

Monolithic:

```
DCIM_iDRACCardView
FQDD = iDRAC.Embedded.1-1
FirmwareVersion = 1.00.00
GUID = 3132334f-c0b7-3480-3510-00364c4c454
IPMIVersion = 2.0
InstanceId = iDRAC.Embedded.1-1#IDRACinfo
LANEnabledState = 1
LastSystemInventoryTime = 20120302092309.000000+000
LastUpdateTime = 20120305233206.000000+000
Model = Enterprise
PermanentMACAddress = 78:2b:cb:54:54:11
ProductDescription = This system component provides a complete set of remote management functions for Dell PowerEdge servers
SOLEnabledState = 1
```

Modular:

```
DCIM_iDRACCardView
FQDD = iDRAC.Embedded.1-1
```
FirmwareVersion = 1.00.00
GUID = 3132334f-c0b7-3480-3510-00364c4c454
IPMIVersion = 2.0
InstanceID = iDRAC.Embedded.1-1#iDRACinfo
LANEnabledState = 1
LastSystemInventoryTime = 20120121022852.000000+000
LastUpdateTime = 20120124015120.000000+000
Model = Express for Blades
PermanentMACAddress = d0:67:e5:f4:2f:97
ProductDescription = This system component provides a complete set of remote management functions for Dell PowerEdge servers
SOLEnabledState = 1

3.9 **CIM Query Language (CQL) Filters**

The CIM Query Language (CQL) is a query language for the Common Information Model (CIM) standard from the Distributed Management Task Force (DMTF). It was designed to perform queries against the CIM objects in a database.

3.9.1 **CQL filter benefits**

The two most distinct advantages of using CQL filters when performing enumerations are:

- The response time of enumerations will be accelerated as only the desired data is returned, not the full data set.
- The workload on the network will be decreased as less bandwidth will be consumed per enumeration as the amount of data being returned is less. This is more applicable to networks that may have many systems that performing enumerations at or about the same time.

3.9.2 **How to perform CQL enumerations using RECITE**

Section 1.2 describes how to setup and run scripts using the RECITE PYTHON environment. Running CQL filters requires running the CQL command directly from the RECITE command line. No scripts exist because of the infinite number of use cases.

3.9.3 **CQL filter example that enumerates all NIC attributes for a particular port/partition**
The FQDD in the example below will be unique to a particular user’s system. The NIC FQDDs of the system can be obtained by running the GetNICViews() command from the RECITE command line.

```
GetNICAttributes -cql="select * from DCIM_NICAttribute where FQDD='NIC.Integrated.1-1-1'
```

3.9.4  **CQL filter example that enumerates all iDRAC attributes with a particular GroupDisplayName and AttributeName**

The GroupDisplayName and the AttributeName in the example below will be unique to a particular user’s system.

```
GetiDRACCardAttributes -cql="select * from DCIM_iDRACCardAttribute where GroupDisplayName='iDRAC Users' and AttributeName='UserName'
```

3.9.5  **CQL filter example that enumerates all BIOS attributes where the IsReadOnly flag is set to TRUE**

The expression below return attributes that have the IsReadOnly=TRUE flag set as the CurrentValue.

```
GetBIOSEnumerations -cql="select CurrentValue from DCIM_BIOSEnumeration where IsReadOnly=TRUE"
```

3.10  **Ordering of iDRAC Attributes to set or apply (11Generation vs 12Generation)**

Users need to correctly set the order in which iDRAC, System, and LC attributes are applied. Incorrect ordering of attributes may result in an error, if dependencies are violated. The DisplayOrder field of each attribute along with the applicable references in profiles, provide direction as to the appropriate ordering.

An example would be to create an iDRAC user account. On 11G systems, the iDRAC would automatically re-order the the attributes before setting/applying them to create and enable a user account. However, due to the expansion of attributes in 12G systems as well as to avoid the anticipating the user’s intent, re-ordering of attributes was removed.

The correct order for setting iDRAC attributes when enabling a user account on both 11G and 12G is as follows:
1. Username
2. Password
3. <other attributes>

This ordering is applicable to both ApplyAttributes() and SetAttributes() iDRAC methods. Use the reverse order when clearing/disabling the account.

### 3.11 How to Determine if Server is 11Generation vs 12Generation

In order to determine if a server is 11G or 12G irrespective of the license present on the server, the recommendation is to look at “LifecycleControllerVersion” property from DCIM_SystemView. If this property is not shown or the value is 1.x.y then it is 11G system. If the value is 2.x.y then it is 12G.

Section 31.4 of this document describes the workflow for enumerating the DCIM_SystemView class.

### 3.12 Using Special Characters in Usernames

For 11G systems, usernames may not contain the characters: <, >, ‘, /.

For 12G systems, usernames may not contain the characters: /,\, @, ., !.

When an iDRAC user has angle brackets for the username or password, they must use double quotes around the brackets when trying to execute any WSMAN commands.

### 3.13 Obtaining Updated System Inventory

Use the following procedure below to refresh stale inventory or to ensure the inventory has the most up to date information. Collect System Inventory on Restart (CSIOR) is the mechanism which checks and updates the inventory. CSIOR is run, when enabled, during the boot process.

**Applies to:** LC1.3.0+

A) [LC1.5.0+] The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) Ensure CSIOR attribute is enabled
   - See Section 31.3 to Check and enable Collect System Inventory on Restart (CSIOR)
C) Power on or reboot system
   - NOTE: If an operating system has been installed, the system will boot into it. It may be desired to wait until the OS boot is complete before performing a graceful shutdown.

D) [LC1.5.0+] The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSSstatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.
   - When the system is ready, the inventory is updated
3.14 How to Determine if the System is Blade, Tower, or Rack

Dell has been officially using “M”, “T” and “R” letters in the Model name to distinguish between “Modular”, “Tower” and “Rack” server respectively for the past couple of generations.

Getting the SystemGeneration attribute can be achieved by viewing the SystemView class. Section 31.4 of this document describes the workflow for enumerating the DCIM_SystemView class.

Examples below:
- 11G Blades: M610, M710, M910
- 12G Blades: M420, M520, M820
- 11G Towers: T310, T410, T610
- 12G Towers: T320, T420, T620
- 11G Racks: R310, R410, R610
- 12G Racks: R320, R520, R820

3.15 Getting the SystemID, Model, and more

Section 31.4 of this document describes the workflow for enumerating the DCIM_SystemView class. Example data returned is shown below.

DCIM_SystemView
AssetTag = tag
BIOSReleaseDate = 08/20/2012
BIOSVersionString = 1.3.5
BaseBoardChassisSlot = NA
BatteryRollupStatus = 1
BladeGeometry = 4
BoardPartNumber = 0N051FX02
BoardSerialNumber = CN13740920003M
CMCIP
CPLDVersion = 0.4.7
CPURollupStatus = 1
ChassisName = Main System Chassis
ChassisServiceTag = S78FGH5
ChassisSystemHeight = 1
ExpressServiceCode = 61387326761
FQDD = System.Embedded.1
FanRollupStatus = 3
HostName
InstanceID = System.Embedded.1
LastSystemInventoryTime = 20130206014757.000000+000
3.16 **Http, CIFS, NFS, tftp, ftp Formatting**

Various protocols are required for methods accessing and/or writing to network shares. The format for these protocols are summarized below.

**HTTP Format:**

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

**CIFS or NFS Format:**

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

**TFTP or FTP Format:**
4 Workflows

The Best Practice Guide provides the detailed step-by-step Lifecycle Controller WSMAN API interactions and algorithmic descriptions needed to implement various system management workflows.

4.1 RAID stacking: ResetConfig, CreateVD, assign HotSpares

This workflow stacks multiple RAID operations together and is applied immediately using the TIME_NOW parameter, which requires one reboot operation. The workflow deletes existing virtual disks and unassigns all hotspares before creating a single virtual disk, a dedicated hotspare, and a global hotspare. Following completion of the reboot, the new virtual disk and hotspare results will be verified. Approximate time for completion on a 12G system is 15 minutes.

**Applies to:** LC1.5.1+

**Prerequisites for script:**

- Set FQDD of desired RAID controller by editing the following script
- Applicable RAID controller and hard drives

**Script:** RAIDstacking.TIME_NOW.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.

B) [LC1.5.1 only] Disable CSIOR (Collect System Inventory on Restart).

**NOTE:** On 11G systems, CSIOR must be disabled to circumvent a sync behavior that prohibits successful RAID stacking.

a. SetLCAttribute(): Sets attribute to be configured [ReturnValue=0]
b. CreateConfigJob(): Creates jobID and applies configuration [ReturnValue=4096]

C) ENUMERATE the DCIM_ControllerView class to find RAID controller’s instanceID & FQDD (They are often identical.) See section 2.2 for a definition of ENUMERATE.

a. Integrated RAID card example is ”RAID.Integrated.1-1”
b. External RAID card example is ”RAID.Slot.1-1”

D) ResetConfig(): Delete all virtual disks and unassign all HotSpare physical disks. [ReturnValue=0]

E) CreateVirtualDisk(): RAID 1 on physical disk 0 & 1, for example. [ReturnValue=0].
F) AssignSpare(): Create dedicated hotspare using Create VD instanceID [ReturnValue=0].

G) AssignSpare(): Create global hotspare [ReturnValue=0].

H) CreateRAIDConfigJob(): Apply steps D) – F) [ReturnValue=4096].

I) Poll jobstatus for Completed: GET the InstanceID of from H). See section 2.3 for a definition of GET.

J) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSSstatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

   The GetRSSstatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

K) ENUMERATE the DCIM_VirtualDiskView class to ensure successful virtual disk creation. See section 2.2 for a definition of ENUMERATE.

   a. RAIDTypes parameter will be 4, for a RAID 1 configuration
   b. PhysicalDiskIDS parameter will list physical disks used

L) ENUMERATE the DCIM_PhysicalDiskView class to ensure successful hotspare assignments. See section 2.2 for a definition of ENUMERATE.

   a. HotSpareStatus parameter of 2, indicates global hotspare
   b. HotSpareStatus parameter of 1, indicates dedicated hotspare

   NOTE: H200 controller is unique in that it always returns 2 for both dedicated and global hotspares

4.2 RAID Stacking with BIOS Attributes Using Setupjobqueue

This workflow stacks multiple RAID operations together along with some BIOS attributes, which requires one reboot operation using setupjobqueue. The workflow deletes existing virtual disks, and unassigns all hotspares before creating a single virtual disk and a dedicated hotspare. Multiple EmbNIC BIOS attributes are also set to Enabled. Following completion of the reboot, the new virtual disk, hotspare, and BIOS attributes results will be verified. Approximate time for completion on a 12G system is 15 minutes.

Applies to: LC1.5.1+

Prerequisites for script:

- Set FQDD of desired RAID controller by editing the following script
- Applicable RAID controller and hard drives

Script: RAIDstacking_BIOS_setupjobqueue.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSSstatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
B) [LC1.5.1 only] Disable CSIOR (Collect System Inventory on Restart).
   NOTE: On 11G systems, CSIOR must be disabled to circumvent a sync behavior that
   prohibits successful RAID stacking.
   a. SetAttribute(): Sets attribute to be configured [ReturnValue=0]
   b. CreateConfigJob(): Creates jobId and applies configuration [ReturnValue=4096]

C) ENUMERATE the DCIM_ControllerView class to find RAID controller’s instanceID & FQDD
   (they are often identical.) See Section 2.2 for a definition of ENUMERATE.
   a. Integrated RAID card example is "RAID.Integrated.1-1"
   b. External RAID card example is "RAID.Slot.1-1"

D) ResetConfig(): Delete all virtual disks and unassign all HotSpare physical disks.
   [ReturnValue=0].

E) CreateVirtualDisk(): RAID 1 on physical disk 0 & 1, for example. [ReturnValue=0].

F) AssignSpare(): Create dedicated hotspare using Create VD instanceID [ReturnValue=0].

G) CreateRAIDConfigJob(): Apply steps D) – F) without reboot type, without UntilTime, and
   without ScheduledStartTime parameter TIME_NOW. [ReturnValue=4096].

H) SetAttribute(): Set BIOS attribute EmbNic1Nic2 to Enabled [ReturnValue=0]

I) CreateBIOSConfigJob(): Apply step H) without reboot type, without UntilTime, and
   without ScheduledStartTime parameter TIME_NOW. [ReturnValue=4096]

J) CreateRebootJob(): Pass RebootJobType of 3 parameter
   1 = PowerCycle
   2 = Graceful reboot without forced shutdown
   3 = Graceful reboot with forced shutdown

K) SetupJobQueue(): Use RAID JID(G), BIOS JID(J), and reboot RID(K) [ReturnValue=0]

L) Poll jobstatus for Completed: GET the InstanceID of from G) or J). See section 2.3 for a
   definition of GET.

M) The Lifecycle Controller remote service must be in a “ready” state before executing any
   other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may
   be used depending on the version of the LC Management registered profile.
   The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while
   the GetRemoteServicesAPISStatus() can just poll for ‘ready.’

N) ENUMERATE the DCIM_VirtualDiskView class to ensure successful virtual disk creation.
   See section 2.2 for a definition of ENUMERATE.
   a. RAIDTypes parameter will be 4, for a RAID 1 configuration
   b. PhysicalDiskIDS parameter will list physical disks used

O) ENUMERATE the DCIM_PhysicalDiskView class to ensure successful hotspare assignments.
   See section 2.2 for a definition of ENUMERATE.
a. HotSpareStatus parameter of 2, indicates global hotspare
b. HotSpareStatus parameter of 1, indicates dedicated hotspare

NOTE: H200 controller is unique in that it always returns 2 for both dedicated and global hotspares

P) ENUMERATE the DCIM_BIOSEnumeration class to ensure BIOS settings were correctly set. See section 2.2 for a definition of ENUMERATE.

4.3 Boot to Network ISO
This workflow boots the host system from an image on a network share. The workflow first removes any existing driver packs or existing attached OS, then gets available OS drivers, unpacks the desired set of drivers, and boots from an image on a network share. Approximate time for completion on a 12G system is 20-40 minutes depending on the size of the driver pack that will be unpacked and the speed of the network to boot image.

Applies to: LC1.3.0+

Prerequisites for script:
- Place applicable ISO image in applicable network share
- Set script variables by editing script

Script: BootToNetworkISO.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

NOTE: GetRemoteServicesAPIStatus() will return “not ready” if drivers or an ISO is already attached.

B) DetachDrivers(): Ensures any drivers are detached.

C) DetachISOImage(): Ensures all images are detached.

D) GetDriverPackInfo(): Displays available OS drivers. This is only required for end to end OS deployment.

E) UnpackAndAttach(): Unpacks and attaches desired driver pack. The resulting concrete job is invoked immediately. This is only required for end to end OS deployment.

F) Poll concrete job until ‘Success’.

G) BootToNetworkISO(): The resulting concrete job is invoked immediately.

H) Poll concrete job until ‘Success’.

NOTE: OS is still booting at this point, so sleep to allow completion. Steps I) through J) are providing when the BootToNetwork image is no longer desired.

I) DetachDrivers(): [ReturnValue=0].

J) DetachISOImage(): [ReturnValue=0].
Lifecyle Controller Integration—Best Practices Specification

K) RequestMonoSystemStateChange(): [ReturnValue=0].

   NOTE: Modular systems (i.e. M610, M710, etc.) use RequestModSystemStateChange().

4.4 Boot to ISO from vFlash

   This workflow boots the host system from an ISO image located on the vFlash. The workflow first removes any existing driver packs or existing attached OS, then gets available OS drivers, unpacks the desired set of drivers, and boots from an image on the vFlash. Approximate time for completion on an 12G system is 20-40 minutes depending on the size of the driver pack to unpack.

   Applies to: LC1.3.0+

   Prerequisites for script:
   - Place applicable ISO image in applicable network share
   - Set script variables by editing script

   Script: BootToVFlash.win

   A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

      NOTE: GetRemoteServicesAPIStatus() will return “not ready” if drivers or an ISO is already attached.

   B) DetachDrivers(): Ensures any previous drivers are detached.

   C) DetachISOImage(): Ensures all previous images are detached.

   D) DetachISOFromVFlash(): Ensures all previous images are detached.

   E) DeleteISOFromVFlash(): Ensures all previous images are deleted.

   F) DownloadISOToVFlash(): Download desired image from network to vFlash.

   G) Poll concrete job until ‘Success’.

   H) GetDriverPackInfo(): Displays available OS drivers. This is only required for end to end OS deployment.

   I) UnpackAndAttach(): Unpacks and attaches desired driver pack. The resulting concrete job is invoked immediately. This is only required for end to end OS deployment.

   J) Poll concrete job until ‘Success’.

   K) BootToISOFromVFlash(): The resulting concrete job is invoked immediately.

   L) Poll concrete job until ‘Success’.

      NOTE: OS boot is complete at this point, sleep 600 seconds to allow for completion. Steps M) through P) are providing when the BootToNetwork image is no longer desired.

   M) DetachDrivers(): [ReturnValue=0].
N) DetachISOFromVFlash(): [ReturnValue=0].
O) DeleteISOFromVFlash(): [ReturnValue=0].
P) RequestMonoSystemStateChange(): Reboot to finish removal of OS [ReturnValue=0].

NOTE: Modular systems (i.e. M610, M710, etc.) use RequestModSystemStateChange().

4.5 Set Hard Disk Drive to ‘first’ in Boot Order
This workflow will set the hard drive(c:) to the top of the boot order. If the BootMode is set to UEFI, it will set it to BIOS. Following completion of the reboot, the hard drive will be enabled and set to boot first. Approximate time for completion that one can expect to encounter on an 12G system is 15-25 minutes, depending on whether an additional reboot is needed to set the BootMode parameter.

Applies to: LC1.4.0+
Prerequisites for script: None
Script: Set_HD_Boot.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) Change BootMode to BIOS, if current value is UEFI.
   a. SetAttribute(): Sets attribute to be configured [ReturnValue=0]
   b. CreateBIOSConfigJob(): Creates jobID and applies configuration [ReturnValue=4096]

C) GetBootConfigSettings(): ENUMERATE the DCIM_BootConfigSetting class to identify the ElementName field containing BootSeq and corresponding InstanceID (IPL or UEFI). See section 2.2 for a definition of ENUMERATE.

   ElementName = Hard drive C: BootSeq

D) GetBootSourceSettings(): ENUMERATE the DCIM_BootSourceSetting class. See section 2.2 for a definition of ENUMERATE.
   a. The CurrentAssignedSequence attribute of each instance defines the instance’s place in the zero based indexed boot sequence
   b. The CurrentEnabledStatus attribute defines whether the boot source, such as the hard drive, is enabled
   c. If the current sequence is 0 and the status is enable, skip to the end

E) ChangeBootOrderByInstanceID(): using instanceID = IPL [ReturnValue=0]
F) ChangeBootSourceState(): using instanceID = IPL and EnabledState=1 [ReturnValue=0]

G) Poll jobstatus for Completed: GET the InstanceID of from E). See section 2.3 for a definition of GET.
H) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

   The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

I) ENUMERATE the DCIM_BootSourceSetting class. See section 2.2 for a definition of ENUMERATE.

   a. The CurrentAssignedSequence of the “Hard drive C" should be 0
   b. The CurrentEnabledStatus of the “Hard drive C" should be 1

4.6 Export (backup) Image to vFlash

   This workflow performs a backup, or export operation, which saves the image to the vFlash. The TIME_NOW parameter is passed, which invokes the operation immediately. Approximate time for completion on an 12G system is 20-50 minutes depending on the system configuration.

   **Applies to:** LC1.5.0+

   **Prerequisites for script:**

   - Valid and enabled vFlash card for licensing
   - [optional] Change passphrase by editing script

   **Script:** Backup_vFlash.win

   A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

   B) BackupImage(): Performs backup operation [ReturnValue=4096].

   C) Poll jobstatus for Completed: GET the InstanceID of from B). See section 2.3 for a definition of GET.

      **NOTE:** The available space on the SD card will be reduced by 384MB upon completion of successful backup.

4.7 Export (backup) Image to CIFS or NFS Share

   This workflow performs a backup, or export operation, which saves the image to a CIFS or NFS share. The TIME_NOW parameter is passed, which invokes the operation immediately. Approximate time for completion on a 12G system is 20-50 minutes depending on the system configuration.

      **NOTE:** The export operation will overwrite an existing backup image on a network share if an identical name is used.

   **Applies to:** LC1.5.0+
Prerequisites for script:

- Valid and enabled vFlash card for licensing
- Set script variables by editing script

Script: Backup_CIFS_NFS.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) BackupImage(): Performs backup operation [ReturnValue=4096].

C) Poll jobstatus for Completed: GET the InstanceID of from B). See section 2.3 for a definition of GET.

4.8 Automatic Backup (12th Generation and Later Version of Servers Only)

This workflow creates a recurring schedule to perform automatic Backup Server profile and export to vFlash or CIFS or NFS share. At the specified schedule, the server profile is backed up and the backup image is exported to the specified target. vFlash can hold only one image at any time. For CIFS or NFS shares, the exported images are automatically renamed and archived. Up to 50 archived images are stored on the CIFS or NFS shares.

Applies to: LC2 1.3.0+ and iDRAC 1.50.50 +

Prerequisites for script:

- A network share (CIFS/NFS) to export the backup images or a vflash card inserted on the system slot
- A software license for 12th Generation Dell PowerEdge servers
- Set script variables by editing script/XML file

Script: Auto_Update.win

A) The Lifecycle Controller remote service must be in the “ready” state before running any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) SetLCAttribute(): Enable the “Automatic Backup Feature” attribute using the SetLCAttribute() function with AttributeName = “Automatic Backup Feature”

C) CreateConfigJob(): Create a Config job and apply this attribute.

D) SetBackupSchedule(): Set the schedule for the automatic backup using the SetBackupSchedule() function. Auto_Backup.xml file provides the input parameters for setting the schedule of the Automatic Backups.

E) GetBackupSchedule(): Get the schedule for the automatic backup using the GetBackupSchedule() function.

F) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
Note: The Automatic Backup schedule can be cleared by using the ClearBackupSchedule() function. The Automatic Backup feature can be disabled by setting the “Automatic Backup Feature” attribute to “Disabled” using the SetLCAttribute() function.

4.9 **Import (restore) Image from vFlash**
This workflow performs a restore, or import operation, which restores the image from the vFlash. The TIME_NOW parameter is passed, which invokes the operation immediately. Approximate time for completion on an 12G system is 30-60 minutes depending on the system configuration.

**Applies to:** LC1.5.0+

**Prerequisites for script:**
- Valid and enabled vFlash card for licensing
- Backup image on SD card (vFlash)
- Passphrase required if backup image used passphrase, edit script

**Script:** Restore_vFlash.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) RestoreImage(): Performs restore operation [ReturnValue=4096].

C) Poll jobstatus for Completed: GET the InstanceID of from B). See section 2.3 for a definition of GET.

4.10 **Import (Restore) Image from CIFS or NFS Share**
This workflow performs a restore, or import operation, which restores an image from the either a CIFS or NFS share. The TIME_NOW parameter is passed, which invokes the operation immediately. Approximate time for completion on an 12G system is 30-60 minutes depending on the system configuration and network.

**Applies to:** LC1.5.0+

**Prerequisites for script:**
- Valid and enabled vFlash card with existing image
- Set script variables by editing script

**Script:** Restore_CIFS_NFS.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
B) RestoreImage(): Performs restore operation [ReturnValue=4096].

C) Poll jobstatus for Completed: GET the InstanceID of from B). See section 2.3 for a definition of GET.

4.11 iDRAC Firmware DUP uUpdate from CIFS or TFTP Share

This workflow performs an update of the iDRAC firmware from a DUP by first downloading the DUP to the system, then applying the update. The update of the iDRAC firmware will be invoked after being scheduled using SetupJobQueue. Approximate time for completion is 30-60 minutes depending on the system configuration and network.

**Applies to**: LC1.3.0+

**Prerequisites for script**:

- Desired DUP must be present on network share
- Set script variables by editing script

**Script**: iDRAC_update.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSoftwareIdentities(): ENUMERATE the DCIM_SoftwareIdentity class to list the firmwares on the system. See section 2.2 for a definition of ENUMERATE.

C) Search the results from B) for:

   - [LC1.5.0/LC1.5.1] "ElementName = iDRAC6" and note the accompanying instanceID to be used in D).
   - [LC2.1.0] "ElementName = Integrated Dell Remote Access Controller" and note the accompanying instanceID to be used in D).

   Use the Software Inventory registered profile version to determine the applicable string to search for.

D) InstallFromURI(): Invokes firmware update operation [ReturnValue=4096].

E) CreateRebootJob(): Pass parameter RebootJobType of value 3.

   1 = PowerCycle
   2 = Graceful reboot without forced shutdown
   3 = Graceful reboot with forced shutdown

F) SetupJobQueue(): Use JID(D) and reboot RID(E) [ReturnValue=0]; The StartTimeInterval parameter is set to TIME_NOW, meaning the operations will be invoked immediately.

G) Poll RID jobstatus for Reboot Completed: GET the InstanceID of from E). See section 2.3 for a definition of GET.

H) Poll JID jobstatus for Completed: GET the InstanceID of from D). See section 2.3 for a definition of GET.
I) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

4.12 BIOS Firmware DUP Update from CIFS or TFTP Share

This workflow performs an update of the BIOS firmware from a DUP by first downloading the DUP to the system, then applying the update. The update of the BIOS firmware will be invoked after being scheduled using SetupJobQueue. Approximate time for completion is 30-60 minutes depending on the system configuration and network.

Applies to: LC1.3.0+

Prerequisites for script:

- Desired DUP must be present on network share
- Set script variables by editing script

Script: BIOS_update.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSoftwareIdentities(): ENUMERATE the DCIM_SoftwareIdentity class to list the firmwares on the system. See section 2.2 for a definition of ENUMERATE.

C) Search the results from B) for “ElementName = BIOS” and for ”Status = Installed”, then note the accompanying instanceID to be used in D)

D) InstallFromURI(): Invokes firmware update operation [ReturnValue=4096]

E) CreateRebootJob(): Pass parameter RebootJobType of value 3
   
   1 = PowerCycle
   2 = Graceful reboot without forced shutdown
   3 = Graceful reboot with forced shutdown

F) SetupJobQueue(): Use JID(D) and reboot RID(E) [ReturnValue=0]; The StartTimeInterval parameter is set to TIME_NOW, meaning the operations will be invoked immediately

G) Poll RID jobstatus for Reboot Completed: GET the InstanceID from E). See section 2.3 for a definition of GET.

H) Poll JID jobstatus for Completed: GET the InstanceID from D). See section 2.3 for a definition of GET.

I) [LC1.5.0/LC1.5.1] Sleep for 5 minutes to allow reboot, POST, and CSIOR to complete

See Appendix 33.4.3 and 33.4.7 for more information about POST and CSIOR

J) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
4.13 **USC Firmware DUP Update from CIFS or TFTP Share**

This workflow performs an update of the USC (LC) firmware from a DUP by first downloading the DUP to the system, then applying the update. By design, the update of the USC firmware will be invoked immediately following download completion and cannot be scheduled for a later time. Approximate time for completion is 30-60 minutes depending on the system configuration.

**Applies to:** LC1.3.0+

**Prerequisites for script:**

- Desired DUP must be present on network share
- Set script variables by editing script

**Script:** USC_LC_update.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSoftwareIdentities(): ENUMERATE the DCIM_SoftwareIdentity class to list the firmwares of the system. See section 2.2 for a definition of ENUMERATE.

C) Search the results from B) for “ElementName = Dell Lifecycle Controller” and note the accompanying instanceID to be used in D). There may be additional characters and numbers after the substring “Controller”.

D) InstallFromURI(): Invokes firmware update operation [ReturnValue=4096]

   **NOTE:** The USC update is applied immediately, and cannot be scheduled for a later time.

E) Poll jobstatus for Completed: GET the InstanceID of from D). See section 2.3 for a definition of GET.

F) RequestiDRACStateChange(): Must reset idrac for changes to take effect [ReturnValue=0]

G) [LC1.5.0/LC1.5.1] Sleep for 10 minutes to allow reboot, POST, and CSIOR to complete

   See Appendix 33.4.3 and 33.4.7 for more information about POST and CSIOR

H) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

4.14 **Automatic Firmware Update (12th Generation and Later Version of Servers Only)**

This workflow creates a recurring schedule to perform multiple firmware updates by specifying a network repository that contains a catalog of available updates and the scheduling parameters. At the specified schedule, all applicable updates contained in the repository will be applied to the system.
**Applies to:** LC2 1.3.0+ and iDRAC 1.50.50 +

**Prerequisites for script:**
- A network share (CIFS/NFS) to access the repository of firmware updates
- A software license for 12th Generation Dell PowerEdge servers
- Set script variables by editing script/XML file

**Script:** Auto_Update.win

A) The Lifecycle Controller remote service must be in a “ready” state before running any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.

B) SetLCAttribute(): Enable the “Automatic Update Feature” attribute using the SetLCAttribute() function with AttributeName = “Automatic Update Feature”

C) CreateConfigJob(): Create a Config job and apply this attribute.

D) SetUpdateSchedule(): Set the schedule for the automatic update using the SetUpdateSchedule() function. Auto_Update.xml file provides the input parameters for setting the schedule of the Automatic Updates.

E) SHENOY TODO: Does this create a job ID? Check with Hari

F) GetUpdateSchedule(): Get the schedule for the automatic update using the GetUpdateSchedule() function

G) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.

Note: The Automatic Update schedule can be cleared by using the ClearUpdateSchedule() function. The Automatic Update feature can be disabled by setting the “Automatic Update Feature” attribute to “Disabled” using the SetLCAttribute() function.

### 4.15 Update from Repository (12th Generation and Later Version of Servers Only)

This workflow allows for update of firmwares using a custom repository created using the Dell Repository Manager.

**Applies to:** LC2 1.3.0+ and iDRAC 1.50.50 +

**Prerequisites for script:**
- A software license for 12th Generation Dell PowerEdge servers
- Set script variables by editing script/XML file
- A network share (CIFS/NFS) to access the repository of firmware updates
- A catalog file for the DUPs

**Script:** RepoUpdate.win

A) The Lifecycle Controller remote service must be in a “ready” state before running any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.
B) InstallFromRepository(): Point to a network share where a catalog file is available and initiate firmware updates from this repository.

C) SHENOY TODO: the rest of the content

D) The Lifecycle Controller remote service must be in a “ready” state before running any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

4.16 Firmware Rollback (12th Generation and Later Version of Servers Only)

This workflow allows a firmware to be rolled back to a previous version of the firmware provided that a previous version exists on the rollback partition.

Applies to: LC2 1.3.0+ and iDRAC 1.50.50 +

Prerequisites for script:

- A software license for 12th Generation Dell PowerEdge servers
- Set script variables by editing script/XML file
- Devices must have previous/available firmware in the Rollback partition.
- User should have “Server Control” privilege except for iDRAC rollback.

Script: Rollback.win

E) The Lifecycle Controller remote service must be in a “ready” state before running any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

F) GetSoftwareIdentities(): Find all versions of the device firmwares. Firmwares which have Installed and Available versions can be rolled back to the Available versions. Firmwares which have only Installed versions cannot be rolled back.

G) Select the Instance ID of the firmware to be rolled back from the output of step B.

H) Invoke the InstallFromSoftwareIdentity() method of the DCIM_SoftwareInstallationService class with Rollback.xml input file. A Job ID is returned by this method.

I) Poll JID jobstatus for Completed: GET the InstanceID from D). See section 2.3 for a definition of GET.

J) The Lifecycle Controller remote service must be in a “ready” state before running any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

4.17 Remote Diagnostics (12th Generation and Later Version of Servers Only)

This workflow provides for remote invocation and execution of Hardware Diagnostics on a Dell PowerEdge Server and exporting the results of the diagnostics execution to a CIFS/NFS share.

Applies to: LC2 1.3.0+ and iDRAC 1.50.50 +

Prerequisites for script:
- A software license for 12th Generation and later versions of Dell PowerEdge servers
- Set script variables by editing script/XML file
  The diagnostics partition must be populated by installing a Diagnostics DUP
  A CIFS/NFS share for exporting the results of the Diagnostics execution

**Script:** RemoteDiagnostic.win

A) The Lifecycle Controller remote service must be in a “ready” state before running any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) RemoteDiagnostics(): This function invokes remote diagnostics and returns a job ID. The remote_diagnostics.xml file is used to provide input parameters. The scheduled start time, the diagnostics mode and the reboot job type can be specified via this XML file.

C) Poll JID jobstatus for Completed: GET the InstanceID from D). See section 2.3 for a definition of GET.

D) ExportDiagResults(): This function causes the results of the diagnostics run to be exported to a CIFS/NFS share specified in the Export_DiagResults.xml file. This returns a Job ID

E) Poll JID jobstatus for Completed: GET the InstanceID from D). See section 2.3 for a definition of GET.

F) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

---

**4.18 PXE Boot using Embedded NICs (11G only)**

This workflow performs a reboot of the host OS into the PXE boot configuration by first setting the embNic1Nic2 parent attributes and embNic1 and embNic2 child attributes. Approximate time for completion on an 11G system is 20-40 minutes depending on the system configuration.

**Applies to:** LC1.5.1+

**Prerequisites for script:** None

**Script:** PXEboot_NIC1_NIC2.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) Enable CSIOR (Collect System Inventory on Restart)
   a. SetAttribute(): Sets attribute to be configured [ReturnValue=0]
   b. CreateConfigJob(): Creates jobID and applies configuration [ReturnValue=4096]

C) Call subroutine sub_setEmbNICs_NIC1_NIC2.win to perform the following:
   a. GetBIOSEnumerations(): Enumerate the DCIM_BIOSEnumeration to obtain the current values of EmbNic attributes
   b. DeletePendingBIOSConfiguration(): Ensures there is no other pending BIOS configuration
   c. SetAttribute(): Set parent attribute EmbNic1Nic2 to DisabledOS [ReturnValue=0]
d. SetAttribute(): Set child attributes EmbNic1 and EmbNic2 to Disabled
   [ReturnValue=0]

e. CreateBIOSConfigJob(): Creates jobID and applies configuration immediately with
   reboot job type of 3 [ReturnValue=4096]

   NOTE: The following RS Status polling for SSIB task

f. The Lifecycle Controller remote service must be in a “ready” state before
   executing any other WSMAN commands. The GetRSStatus() method or the
   GetRemoteServicesAPIStatus() method may be used depending on the version of
   the LC Management registered profile.

   The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’,
   while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

   NOTE: The following RS Status polling is for PXE to be set in the boot list during CSIOR

g. The Lifecycle Controller remote service must be in a “ready” state before
   executing any other WSMAN commands. The GetRSStatus() method or the
   GetRemoteServicesAPIStatus() method may be used depending on the version of
   the LC Management registered profile.

   The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’,
   while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

D) Call subroutine sub_setEmbNICs_NIC1_NIC2.win to perform the following:

a. GetBIOSEnumerations(): Enumerate the DCIM_BIOSEnumeration to obtain the
   current values of EmbNic attributes

b. DeletePendingBIOSConfiguration(): Ensures there is no other pending BIOS
   configuration

c. SetAttribute(): Set parent attribute EmbNic1Nic2 to Enabled [ReturnValue=0]

d. SetAttribute(): Set child attributes EmbNic1 and EmbNic2 to EnabledPxe
   [ReturnValue=0]

e. CreateBIOSConfigJob(): Creates jobID and applies configuration immediately with
   reboot job type of 3 [ReturnValue=4096]

   NOTE: The following RS Status polling for SSIB task

f. The Lifecycle Controller remote service must be in a “ready” state before
   executing any other WSMAN commands. The GetRSStatus() method or the
   GetRemoteServicesAPIStatus() method may be used depending on the version of
   the LC Management registered profile.

   The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’,
   while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

   NOTE: The following RS Status polling is for PXE to be set in the boot list during CSIOR

g. The Lifecycle Controller remote service must be in a “ready” state before
   executing any other WSMAN commands. The GetRSStatus() method or the
   GetRemoteServicesAPIStatus() method may be used depending on the version of
   the LC Management registered profile.
The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

E) Sleep 500 seconds to allow PXE boot to occur. Users would then select applicable PXE boot options before continuing.

Proceed to step F) to disable PXE boot.

F) Call subroutine sub_setEmbNICs_NIC1_NIC2.win to perform the following:
   a. GetBIOSEnumerations(): Enumerate the DCIM_BIOSEnumeration to obtain the current values of EmbNic attributes
   b. DeletePendingBIOSConfiguration(): Ensures there is no other pending BIOS configuration
   c. SetAttribute(): Set parent attribute EmbNic1Nic2 to Enabled [ReturnValue=0]
   d. SetAttribute(): Set child attributes EmbNic1 and EmbNic2 to Enabled [ReturnValue=0]
   e. CreateBIOSConfigJob(): Creates jobID and applies configuration immediately with reboot job type of 3 [ReturnValue=4096]
      NOTE: The following RS Status polling for SSIB task
   f. The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’
      NOTE: The following RS Status polling is for PXE to be set in the boot list during CSIOR
   g. The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

4.19 PXE Boot using Embedded NICs (12G only)
This workflow performs a reboot of the host OS into the PXE boot configuration by first setting the embNic1Nic2 parent attribute and EmbNicPort1BootProto (to Pxe) and EmbNicPort2BootProto (to None) child attributes. Approximate time for completion on a 12G system is 10-30 minutes depending on the system configuration.

Applies to: LC2+

Prerequisites for script: None

Script: PXEboot_NIC1_NIC2_12G.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

   GetRemoteServicesAPIStatus()

B) Enable CSIOR (Collect System Inventory on Restart)
   a. SetAttribute(): Sets attribute to be configured [ReturnValue=0]
   b. CreateConfigJob(): Creates jobID and applies configuration [ReturnValue=4096]

C) Call subroutine sub_setEmbNICs_NIC1_NIC2_12G.win to perform the following:
4.20 Set NIC Attributes and iSCSI boot using setupjobqueue (11G only)

This workflow sets several NIC attributes, including setup of iSCSI boot, by first setting the required attributes for the operation, and then changing the boot order for the NIC.
Approximate time for completion on an 11G system is 15-30 minutes depending on the system configuration.

**Applies to:** LC1.5.0 & LC1.5.1

**Prerequisites for script:** Set script variables by editing script

**Script:** setNICs_iSCSI_boot.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) **ENUMERATE** the DCIM_NICView and DCIM_SoftwareIdentity classes to collect information about the system. See section 2.2 for a definition of **ENUMERATE**.

C) Enable CSIOR (Collect System Inventory on Restart), if not enabled
   a. SetAttribute(): Sets attribute to be configured [ReturnValue=0]
   b. CreateConfigJob(): Creates jobID and applies configuration [ReturnValue=4096]
   c. Poll jobstatus for Completed: GET the InstanceID of from B). See section 2.3 for a definition of GET.

D) **ENUMERATE** the DCIM_NICEnumeration, DCIM_NICString, DCIM_NICInteger, and DCIM_BIOSEnumeration classes to collect information about the system. See section 2.2 for a definition of **ENUMERATE**.

E) SetBIOSAttributes(): Set all the following attributes, if at least one is not set to desired value
   a. EmbNic1Nic2=Enabled
   b. BootMode=BIOS
   c. ProcVirtualization= Enabled
   d. ErrPrompt=Disabled
   e. EmbNic1=Enabled

F) CreateBIOSConfigJob(): Apply step E) with reboot type 3 and ScheduledStartTime parameter of TIME_NOW, which invokes the operation immediately [ReturnValue=4096]

G) Poll jobstatus for Completed: GET the InstanceID of from F). See section 2.3 for a definition of GET.

**NOTE:** The following status polling for SSIB task

F) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

**NOTE:** The following status polling is for subsequent CSIOR

G) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
H) **ENUMERATE** the `DCIM_BootSourceSetting` class to collect information about the system. See section 2.2 for a definition of **ENUMERATE**.

I) `ChangeBootSourceState()`: Loop through boot sources and set their enabled state to zero, except for NIC [ReturnValue=4096]

J) `ChangeBootOrderByInstanceId()`: Set the boot order of the NIC to first (CurrentAssignedSequence = 0) [ReturnValue=0]

K) `setAttribute()`: Set BIOS attributes EmbNic1 to EnablediScsi [ReturnValue=0]

L) `CreateBIOSConfigJob()`: Apply steps J) - L) without reboot, without UntilTime, and without ScheduledStartTime parameter TIME_NOW. [ReturnValue=4096]

M) `setAttribute()`: Set various NIC attributes

N) `CreateNICConfigJob()`: Apply steps N) without reboot, without UntilTime, and without ScheduledStartTime parameter TIME_NOW. [ReturnValue=4096]

O) `CreateRebootJob()`: Pass RebootJobType of 3 parameter
   
   1 = PowerCycle
   2 = Graceful reboot without forced shutdown
   3 = Graceful reboot with forced shutdown

P) `SetupJobQueue()`: Use BIOS JID(L), NIC JID(N), and reboot RID(O) [ReturnValue=0]

Q) Poll jobstatus for Completed: GET the `InstanceId` of from M). See section 2.3 for a definition of **GET**.

R) Poll jobstatus for Completed: GET the `InstanceId` of from O). See section 2.3 for a definition of **GET**.

S) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

### 4.21 iSCSI Boot using NDC/Broadcom (12G only)

This workflow sets several NIC attributes, including setup of iSCSI boot, by first setting the required attributes for the operation, and then changing the boot order for the NIC. Additional detail is provided as setting up iSCSI boot is complex. Approximate time for completion on a 12G system is 15-30 minutes depending on the system configuration.

**NOTE**: Additional details of the coding steps shown below, generational differences, and other iSCSI information can be found in Appendix: iSCSI Boot information.

**Applies to**: LC2+

**Prerequisites for script**:

- Network Daughter Card (NDC) / Broadcom NIC
- This script will only work using an NIC FQDD of NIC.Integrated.1-1 or NIC.Integrated.1-1-1
Only works with first port-partition of 12G NDC (script limitation)
System must be powered off
Edit script to set desired parameters

**Script:** iscsiboot-12gNDC.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   
   GetRemoteServicesAPIStatus():

B) GetBIOSEnumerations: ENUMERATE the `DCIM_BIOSEnumeration` class to collect information about the system. See section 2.2 for a definition of ENUMERATE.

C) GetNICViews: ENUMERATE the `DCIM_NICVIEW` class to collect information about the NIC FQDDs. See section 2.2 for a definition of ENUMERATE.

D) GetBootSourceSettings: ENUMERATE the `DCIM_BootSourceSetting` class to collect information about the NIC FQDDs. See section 2.2 for a definition of ENUMERATE.
   
   - Check whether the FQDD and IPL fields are in the boot order
   - SetNICAttributes(): Set the attribute LegacyBootProto to the value “iSCSI” and the other desired NIC attributes and values
   - CreateNICConfigJob(): Pass RebootJobType=1

E) GetBootSourceSettings(): ENUMERATE the `DCIM_BootSourceSetting` class to collect information about the NICs. See section 2.2 for a definition of ENUMERATE.
   
   - Check the CurrentEnabledStatus to ensure it is enabled

F) ChangeBootSourceState(): Use InstanceID=IPL EnabledState=1 source=(instanceID from D)

G) ChangeBootOrderByInstanceId(): Use InstanceID=IPL source=(instanceID from D)

H) CreateBIOSConfigJob(): Use Target=(BIOS FQDD)

I) Poll jobstatus for Completed: GET the InstanceID of from F). See section 2.3 for a definition of GET.

### 4.22 iSCSI Boot using QLogic (12G only)

This workflow sets several NIC attributes, including setup of iSCSI boot, by first setting the required attributes for the operation, and then changing the boot order for the NIC. Additional detail is provided as setting up iSCSI boot is complex. Approximate time for completion on a 12G system is 15-30 minutes depending on the system configuration.

**NOTE:** Additional details of the coding steps shown below, generational differences, and other iSCSI information can be found in Appendix: iSCSI Boot information.

**Applies to:** LC2+

**Prerequisites for script:**

- QLogic NIC
• Edit script to set desired parameters
• FQDD must be NIC.Integrated.1-1-1 or NIC.Integrated.1-1

**Script:** iscsiboot-12gNDC-qlogic.win

The four high level steps, which may require four reboots, are the following:

1) If the NDC is disabled, 1 job to enable it in bios and reboot
2) Now that the card is present, if iscsi offload on 1-3 is disabled, create a job and reboot to enable it
3) Once iSCSI is enabled, configure iSCSI, but it will not show up in the boot order until after reboot
4) Once it is in the boot order, move it to the top of the HD list

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   - GetRemoteServicesAPIStatus():

B) GetBIOSEnumerations(): **ENUMERATE** the DCIM_BIOSEnumeration class to collect information about the system. See section 2.2 for a definition of **ENUMERATE**.
   - Ensure AttributeName of IntegratedNetwork1 is enabled
   - If it is not enabled, enable it as shown below
   - SetBIOSAttributes()
     - AttributeName=IntegratedNetwork1 AttributeValue=Enabled
     - AttributeName=BootMode AttributeValue=Bios
   - CreateBIOSConfigJob()
     - ScheduledStartTime=TIME_NOW RebootJobType=1
   - Poll jobstatus for Completed: GET the InstanceID of from 2).

C) GetNICViews: **ENUMERATE** the DCIM_NICview class to collect information about the NIC FQDDs. See section 2.2 for a definition of **ENUMERATE**.
   - Check if specified FQDD is present in NICViews, If not, go to NICError

D) GetBootSourceSettings(): **ENUMERATE** the DCIM_BootSourceSetting class to collect information about the boot sources. See section 2.2 for a definition of **ENUMERATE**.
   - Loop through all boot sources, if boot source is IPL entry, set EnabledState=0 unless HD

E) ChangeBootSourceState(): Use InstanceID=IPL EnabledState=0 source=(instanceID from D)

F) GetBootSourceSettings(): **ENUMERATE** the DCIM_BootSourceSetting class to collect information about the boot sources. See section 2.2 for a definition of **ENUMERATE**.
   - Enable the HD boot source
G) ChangeBootSourceState(): Use InstanceID=IPL EnabledState=1 source=(instanceID from F)

H) GetBootSourceSettings(): ENUMERATE the DCIM_BootSourceSetting class to collect information about the boot sources. See section 2.2 for a definition of ENUMERATE.
   • Change NIC boot source

I) GetBootSourceSettings(): ENUMERATE the DCIM_BootSourceSetting class to collect information about the boot sources. See section 2.2 for a definition of ENUMERATE.
   • Check NIC boot order

J) ChangeBootOrderByInstanceID(): Use InstanceID=IPL source=(instanceID from I)
   • SetNICAttributes(): Set the attribute LegacyBootProto to the value “iSCSI” and the other desired NIC attributes and values

K) CreateBIOSConfigJob(): Use Target=(BIOS FQDD)
   • ScheduledStartTime=TIME_NOW RebootJobType=1

L) Poll jobstatus for Completed: GET the InstanceID of from F). See section 2.3 for a definition of GET.

Notes:

1) QLogic will not show up in the boot list until it connects to an iSCSI target. So if iSCSI is misconfigured, or the network is down, it does not show up.
2) RAID and SATA HDs cannot be disabled in the boot list. Either disable the controller, but then they are not available as secondary disks, or move them down in the HD boot list.
3) It is recommended to disable the entire HD list from the boot order until iSCSI is on the top, to prevent it from booting into another HD

4.23 iSCSI boot using Intel (12th Generation only)
This workflow sets several NIC attributes, including setup of iSCSI boot, by first setting the required attributes for the operation, and then changing the boot order for the NIC. Additional detail is provided as setting up iSCSI boot is complex. Approximate time for completion on a 12G system is 15-30 minutes depending on the system configuration.

NOTE: Additional details of the coding steps shown below, generational differences, and other iSCSI information can be found in Appendix: iSCSI Boot information.

Applies to: LC2+

Prerequisites for script:

• Intel NIC
• Edit script to set desired parameters
**Script:** iscsiboot-12gHBA-intel.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   - GetRemoteServicesAPIStatus()

B) GetNICViews: **ENUMERATE** the **DCIM_NICVIEW** class to collect information about the NIC FQDDs. See section 2.2 for a definition of **ENUMERATE**.
   - Check if specified NIC FQDD is present in NICViews

C) GetNICEnumerations: **ENUMERATE** the **DCIM_NICEnumeration** class to collect information about the system. See section 2.2 for a definition of **ENUMERATE**.
   - Check if TcpIpViaDHCP=Enabled and IscsiViaDHCP=Disabled

D) GetBootSourceSettings: **ENUMERATE** the **DCIM_BootSourceSetting** class to collect information about the NIC FQDDs. See section 2.2 for a definition of **ENUMERATE**.
   - Loop through all boot sources, if boot source is IPL entry, set CurrentEnabledStatus=0 unless HD [Steps D)-F)]

E) ChangeBootSourceState(): Use InstanceID=IPL EnabledState=0 source=(instanceID from C)
   - Set CurrentEnabledStatus=1 for NIC FQDD boot source

F) GetBootSourceSettings(): **ENUMERATE** the **DCIM_BootSourceSetting** class to collect information about the NICs. See section 2.2 for a definition of **ENUMERATE**.
   - Check the CurrentEnabledStatus state

G) Configure iSCSI
   - CreateBIOSConfigJob(): Target=(BIOS FQDD)
   - SetNICAttributes(): Target=(NIC FQDD) Set the attribute LegacyBootProto to the value iSCSIPrimary
   - CreateNICConfigJob(): Target=(NIC FQDD)
   - Poll jobstatus for Completed using instanceID from CreateNICConfigJob()

H) Move iSCSI to the top of the HD Boot List by looping through boot sources
   - GetBootSourceSettings()
   - ChangeBootSourceState(): Use InstanceID=IPL EnabledState=1 source=(instanceID from GetBootSourceSettings())
I) GetBootSourceSettings(): ENUMERATE the DCIM_BootSourceSetting class to collect information about the NIC FQDDs. See section 2.2 for a definition of ENUMERATE.

- Loop through boot sources to confirm the NIC FQDD and “BCV” are in an instanceID

J) Set NIC to first in boot order

- GetBootSourceSettings()
- ChangeBootOrderByInstanceID(): Use InstanceID=BCV and source=(instanceID from GetBootSourceSettings())
- CreateBIOSConfigJob(): Use Target=(BIOS FQDD)
- Poll jobstatus for Completed: GET the InstanceID of from BIOS config job

Notes:

1) Intel will not show up in the boot list until it connects to an iSCSI target. So if you misconfigure iSCSI or the network is down it does not show up.
2) You cannot disable RAID and SATA HDs in the boot list. Either disable the controller, but then they are not available as secondary disks or move them down in the HD boot list.
3) It may be a good idea to disable the whole HD list from the boot order until iSCSI is on the top, to prevent it booting into another HD

4.24 IO Identity

This workflow sets IO Identity NIC attributes by first setting the required attributes for the operation, and then applying the changes immediately using the TIME_NOW parameter. Note that these attributes are read-only on some NIC cards, and read-write on others (i.e. Broadcom 57712). Approximate time for completion on a 12G system is 15-30 minutes depending on the system configuration.

NOTE: These attributes can only be set remotely, not locally. Also, if A/C power is lost, these settings will also be lost.

Applies to: LC1.5.0+

Prerequisites for script:

- Broadcom 57712 hardware or QLogic 8262
- Set FQDD of desired NIC, VirtIscsiMacAddr, and VirtMacAddr by editing the following script

Script: IO_Identity.win
A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) ENUMERATE the `DCIM_NICString` class to collect information about the NIC’s attributes and FQDD. See section 2.2 for a definition of ENUMERATE.

C) SetAttribute(): Set various NIC attributes

D) CreateNICConfigJob(): Apply step C) [ReturnValue=4096]

E) Poll jobstatus for Completed: GET the `InstanceID` of from D). See section 2.3 for a definition of GET.

F) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

   The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

   [Pre-LC2] NOTE: The following RS Status polling is for CSIOR, assuming its enabled

G) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

   The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

H) ENUMERATE the `DCIM_NICString` class and ensure the attributes were applied correctly. See section 2.2 for a definition of ENUMERATE.

4.25 Export LC log

This workflow exports the LC log to either an NFS or CIFS share. Approximate time for completion on an 12G system is 5 minutes depending on the system configuration.

Applies to: LC1.5.0+

Prerequisites for script:
- Set script variables by editing script

Script: ExportLCLog.win

A) The Lifecycle Controller remote service must be in a “ready” state before running any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) ExportLCLLog(): Invokes the export operation [ReturnValue=4096]

   NOTE: The user must set applicable IP address, username, password, filename, and workgroup of the network share.
C) Poll jobstatus for Completed: GET the InstanceID of from D). See section 2.3 for a definition of GET.

Note: In iDRAC 1.50.50, there is a provision to export the full Lifecycle log (including all active and archived logs). This allows for exceeding the previously set 64K limit. The function ExportFullLCLLog() can be used to export the full Lifecycle log.

4.26 FCoE Boot using QLogic (12G only)

This workflow sets several NIC attributes, including setup of Fiber Channel over Ethernet (FCoE), by first setting the attributes for the operation and then changing the boot order for the NIC. Approximate time for completion on a 12G system is 20-30 minutes depending on the system configuration.

The four high level steps, which may require 3 reboots, are the following:

1) If the NDC is disabled, 1 job to enable it in bios and reboot
2) Now that the card is present, if FCoE offload on 1-1-4 is disabled, enable it
   Once FCoE is enabled, configure FCoE, but it will not show up in the boot order until after reboot
3) Once it is in the boot order, move it to the top of the HD list

Notes:
1) QLogic will not show up in the boot list until it connects to an FCoE target. So if you misconfigure FCoE or the network is down it does not show up.
2) You cannot disable RAID and SATA HDs in the boot list. Either disable the controller, but then they are not available as secondary disks or move them down in the HD boot list.
3) It may be a good idea to disable the whole HD list from the boot order until FCoE is on the top, to prevent it booting into another HD
4) port settings are configured against partition 1 (NIC.Integrated.1-1-1) but the boot target will show up on partition 4 (BCV:BIOS.Setup.1-1#HddSeq#NIC.Integrated.1-1-4)

Applies to: LC2+

Prerequisites for script:

- QLogic NDC
- Set desired script variables by editing script
- FQDD must be NIC.Integrated.1-1-1 (for port setting) and NIC.Integrated.1-1-4 (for boot target setting)

Script: FCoEboot-12gNDC-qlogic.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
1. System should be power off
2. Clear all unfinished jobs
3. Clear all pending data

B) Check NDC is enabled
1. GetBIOSEnumerations(): ENUMERATE the DCIM_BIOSEnumeration class to collect information about the system. See section 2.2 for a definition of ENUMERATE.

2. Ensure AttributeName of IntegratedNetwork1 is enabled
   If it is not enabled, enable it as shown below
   - SetBIOSAttributes()
     AttributeName=IntegratedNetwork1 AttributeValue=Enabled
   - CreateBIOSConfigJob()
   - ScheduledStartTime=TIME_NOW RebootJobType=1
   - Poll jobstatus for Completed: GET the InstanceID of from 2).

C) CheckConnectFirstFCoETarget(): ENUMERATE the NIC FADD and check if ConnectFirstFCoETarget is enabled, if not, enable ConnectFirstFCoETarget as show below
   - Disable all sources
   - Create BIOS job
   - SetNICAttributes()
     AttributeName=ConnectFirstFCoETarget AttributeValue=Enabled
   - CreateNICConfigJob with RebootJobType=1

D) Configure FCoE
1. Disable all sources
2. Create BIOS job
3. Set Partition Attributes as follows:
   SetNICAttributes() on NIC.Integrated.1-1-4
   AttributeName=FCoEOffloadMode AttributeValue=Enabled
   AttributeName=VirtFiPMacAddr AttributeValue=$VirtFiPMacAddr
   AttributeName=VirtWWN AttributeValue=$VirtWWN AttributeName=VirtWWPN
   AttributeValue=$VirtWWPN AttributeName=MinBandwidth
   AttributeValue=$MinBandwidth AttributeName=MaxBandwidth
   AttributeValue=$MaxBandwidth

4. CreateNICConfigJob()
5. Set Port Attributes as follows:
   SetNICAttributes() on NIC.Integrated.1-1-1
   AttributeName=FirstFCoEWWPNTarget AttributeValue=$FirstFCoEWWPNTarget
   AttributeName=FirstFCoEBootTargetLUN AttributeValue=$FirstFCoEBootTargetLUN

6. CreateNICConfigJob() with RebootJobType=1

E) GetBootSourceSettings(): ENUMERATE the DCIM_BootSourceSetting class to collect information about the boot sources. See section 2.2 for a definition of ENUMERATE.
   Loop through all boot sources, if boot source is IPL entry, set EnabledState=0 unless HD.

F) ChangeBootSourceState(): Use InstanceID=IPL EnabledState=0 source=(instanceID from D)
G) GetBootSourceSettings(): ENUMERATE the `DCIM_BootSourceSetting` class to collect information about the boot sources. See section 2.2 for a definition of ENUMERATE.

H) Enable the HD boot source

I) ChangeBootSourceState(): Use InstanceID=IPL EnabledState=1 source=(instanceID from F) GetBootSourceSettings(): ENUMERATE the `DCIM_BootSourceSetting` class to collect information about the boot sources. See section 2.2 for a definition of ENUMERATE. Change NIC boot source

J) GetBootSourceSettings(): ENUMERATE the `DCIM_BootSourceSetting` class to collect information about the boot sources. See section 2.2 for a definition of ENUMERATE. Check NIC boot order

K) ChangeBootOrderByInstanceID(): Use InstanceID=IPL source=(instanceID from I) SetNICAttributes(): Set the attribute LegacyBootProto to the value “FCoE” and the other desired NIC attributes and values

L) CreateBIOSConfigJob(): Use Target=(BIOS FQDD) ScheduledStartTime=TIME_NOW RebootJobType=1

M) Poll jobstatus for Completed: GET the `InstanceID` of from F). See section 2.3 for a definition of GET.

4.27 **FCoE boot using Intel (12th Generation only)**

This workflow sets several NIC attributes, including setup of Fiber Channel over Ethernet (FCoE), by first setting the attributes for the operation and then changing the boot order for the NIC. Approximate time for completion on a 12G system is 20-30 minutes depending on the system configuration.

**Applies to:** LC2+

**Prerequisites for script:**

- Intel NIC Mezz card
- Set desired script variables by editing script

**NOTES:**

- Make sure Mezz NIC is enabled
  
  If not enabled, enable it and find NIC FQDD, reboot

- Enable ConnectFirstFCoETarget
  
  Check if ConnectFirstFCoETarget is enabled, if not enable it

  Configure FCoE, reboot

- Move FCoE to the top of the HD Boot List
Script: FCoEboot-12g-IntelMezz.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. GetRemoteServicesAPIStatus():
   1. System should be power off
   2. Clear all unfinished jobs
   3. Clear all pending data

B) Check NIC is enabled
   1. GetBIOSEnumerations(): ENUMERATE the DCIM_BIOSEnumeration class to collect information about the system. See section 2.2 for a definition of ENUMERATE.
   2. Ensure AttributeName of Slot2 is enabled
      If it is not enabled, enable it as shown below
      SetBIOSAttributes()
      • AttributeName= Slot2 AttributeValue=Enabled
      • AttributeName=BootMode AttributeValue=Bios
      • CreateBIOSConfigJob()
      • ScheduledStartTime=TIME_NOW RebootJobType=1
      • Poll jobstatus for Completed: GET the InstanceID of from 2).

C) CheckConnectFirstFCoETarget(): ENUMERATE the NIC FADD and check if ConnectFirstFCoETarget is enabled, if not, enable ConnectFirstFCoETarget as show below
   • SetNICAttributes()
      AttributeName=LegacyBootProto AttributeValue=FCoE
      AttributeName=ConnectFirstFCoETarget AttributeValue=Enabled
   • Disable all sources
   • Create BIOS job
   • SetNICAttributes()
      AttributeName=ConnectFirstFCoETarget AttributeValue=Enabled
   • CreateNICConfigJob with RebootJobType=1

D) Configure FCoE
1. Disable all sources
2. Create BIOS job
3. Set Attributes (VLAN etc) as follows
   - `SetNICAttributes()` on NIC.Mezzanine.2B-1
     - `AttributeName=FCoEOffloadMode` `AttributeValue=Enabled`
     - `AttributeName=VirtFIPMacAddr` `AttributeValue=${VirtFIPMacAddr}`
     - `AttributeName=VirtWWN` `AttributeValue=${VirtWWN}`
     - `AttributeName=VirtWWPN` `AttributeValue=${VirtWWPN}`
     - `AttributeName=MinBandwidth` `AttributeValue=${MinBandwidth}`
     - `AttributeName=MaxBandwidth` `AttributeValue=${MaxBandwidth}`
4. `CreateNICConfigJob()`
5. Set Attributes (target) as follows
   - `SetNICAttributes()` on NIC.Mezzanine.2B-1
     - `AttributeName=FirstFCoEWWPNTarget` `AttributeValue=${FirstFCoEWWPNTarget}`
     - `AttributeName=FirstFCoEBootTargetLUN` `AttributeValue=${FirstFCoEBootTargetLUN}`
6. `CreateNICConfigJob()` with `RebootJobType=1`

E) GetBootSourceSettings(): ENUMERATE the `DCIM_BootSourceSetting` class to collect information about the boot sources. See section 2.2 for a definition of ENUMERATE. Loop through all boot sources, if boot source is IPL entry, set `EnabledState=0` unless HD.

F) ChangeBootSourceState(): Use `InstanceID=IPL EnabledState=0 source=(instanceID from D)`

G) GetBootSourceSettings(): ENUMERATE the `DCIM_BootSourceSetting` class to collect information about the boot sources. See section 2.2 for a definition of ENUMERATE.

H) Enable the HD boot source

I) ChangeBootSourceState(): Use `InstanceID=IPL EnabledState=1 source=(instanceID from F)`
   GetBootSourceSettings(): ENUMERATE the `DCIM_BootSourceSetting` class to collect information about the boot sources. See section 2.2 for a definition of ENUMERATE.
   Change NIC boot source

J) GetBootSourceSettings(): ENUMERATE the `DCIM_BootSourceSetting` class to collect information about the boot sources. See section 2.2 for a definition of ENUMERATE.
   Check NIC boot order
K) ChangeBootOrderByInstanceID(): Use InstanceID=IPL source=(instanceID from I)
SetNICAttributes(): Set the attribute LegacyBootProto to the value “FCoE” and the other desired NIC attributes and values

L) CreateBIOSConfigJob(): Use Target=(BIOS FQDD)
ScheduledStartTime=TIME_NOW RebootJobType=1

M) Poll jobstatus for Completed: GET the InstanceID of from F). See section 2.3 for a definition of GET.

4.28 FCoE boot using Broadcom (12G only)
This workflow sets several NIC attributes, including setup of Fiber Channel over Ethernet (FCoE), by first setting the attributes for the operation and then changing the boot order for the NIC. Approximate time for completion on a 12G system is 20-30 minutes depending on the system configuration.

Applies to: LC2+

Prerequisites for script:
- Broadcom Mezz card using slot1
- Set desired script variables by editing script

Script: FCoEboot-12g-Broadcom.win
The four high level steps, which may require 3 reboots, are the follows:

1) If the NIC is disabled, 1 job to enable it in bios and reboot
2) If partition disabled, 1 job to enable partition and reboot
3) Once both NIC card and partition are enabled, set the FCoE related NIC attributes and reboot
4) After that, the NIC will show up in the IPL list, enable it.

A) Check the NIC is enabled
1. GetBIOSEnumerations(): ENUMERATE the DCIM_BIOSEnumeration class to collect information about the system.

2. Ensure AttributeName of $nicAttributeName is enabled
   If it is not enabled, enable it as shown below
   - SetBIOSAttributes()
     AttributeName=$nicAttributeName AttributeValue=Enabled
     AttributeName=BootMode AttributeValue=Bios
CreateBIOSConfigJob()
ScheduledStartTime=TIME_NOW RebootJobType=1
Poll jobstatus for Completed

B) Check partition is enabled, if not enable it as follows
AttributeName=NicPartitioning AttributeValue=Enabled

C) Configure FCoE

1. Disable all sources
2. Create BIOS job
3. Set the following attributes
   LegacyBootProto=FCoE
   ConnectFirstFCoETarget=Enabled
   FCoEOffloadMode=Enabled
   FCoETgtBoot=Enabled
   VirtWWPN=your WWPN address
   FirstFCoEWWPNTarget=your FirstFCoEWWPNTarget address
   FirstFCoEBootTargetLUN=your FirstFCoEBootTargetLUN value

D) GetBootSourceSettings(): ENUMERATE the DCIM_BootSourceSetting class to collect
   information about the boot sources.

E) Loop through all boot sources, until find the NIC, then enable the NIC boot source

4.29 IO Identity for QLogic (12G only)

This workflow sets IO Identity, which are the following: virtual Mac address, virtual iSCSI Mac
address, virtual FIP Mac address, virtual WWN, and virtual WWPN. This workflow is for Qlogic
cards only.

This script has been tested on Qlogic Mezz card, slot 2 with the following setting. Different
cards or different slots require modifications to the script.

- $nic=NIC.Mezzanine.2B-1-1
- $nic2=NIC.Mezzanine.2B-1-2
- $nic3=NIC.Mezzanine.2B-1-3
- $nic4=NIC.Mezzanine.2B-1-4
- $nicAttributeName=Slot2

The script is an example for set IO identity for the following card setting:

- Partition 1: NIC
- Partition 3: iSCSI
- Partition 4: FCoE

Applies to: LC2+
Prerequisites for script:

- QLogic card
- Set desired script variables by editing script

Script: IO_IdentityQlogic.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus():

B) Check NIC is enabled

1. GetBIOSEnumerations(): ENUMERATE the **DCIM_BIOSEnumeration** class to collect information about the system.

2. Ensure AttributeName of $nicAttributeName is enabled
   If not enabled, enable it as shown below
   - SetBIOSAttributes()
     AttributeName=$nicAttributeName  AttributeValue=Enabled
     AttributeName=BootMode AttributeValue=Bios
   - CreateBIOSConfigJob()
   - ScheduledStartTime=TIME_NOW RebootJobType=1
   - Poll job status for Completed.

C) Disable the FlexAddress

- SetLCAttributes AttributeName=VirtualAddressManagement AttributeValue=Console

D) Set IO Identity

- SetNICAttributes()
  - For partition 1: AttributeName=VirtMacAddr AttributeValue=$VirtMacAddr_value
  - For partition 3: AttributeName=VirtIscsiMacAddr AttributeValue=$VirtIscsiMacAddr_value
  - For partition 4:  AttributeName=VirtFIPMacAddr AttributeValue=$VirtFIPMacAddr_value
    AttributeName=VirtWWN AttributeValue=$VirtWWN_value
    AttributeName=VirtWWPN AttributeValue=$VirtWWPN_value
- CreateNICConfigJob() for
  - Partition 1
  - Partition 3
  - Partition 4
  - Then Reboot
4.30 **IO Identity for Broadcom (12G only)**

This workflow sets IO Identity for Broadcom card only.

The script is an example for setting IO Identity for the following set:

- Partition Disabled

**Applies to:** LC2+

**Prerequisites for script:**

- Broadcom card
- Set desired script variables by editing script

**Script:** IO_IdentityBroadcom.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus():

B) Check NIC is enabled

1. GetBIOSEnumerations(): ENUMERATE the `DCIM_BIOSEnumeration` class to collect information about the system.

2. Ensure the NIC is enabled

   If not enabled, enable it as shown below

   - SetBIOSAttributes() Target = (BIOS FQDD)
   - CreateBIOSConfigJob() Target = (BIOS FQDD)
   - ScheduledStartTime=TIME_NOW RebootJobType=1
   - Poll job status for Completed

C) Disable the FlexAddress

   - SetLCAtributes AttributeName=VirtualAddressManagement
   
D) Set IO Identity
- SetNICAttributes() Target = (NIC FQDD)
- CreateNICConfigJob() Target = (NIC FQDD)
- Poll job status for completion using instanceID form CreateNICConfigJob()
4.31 **IO Identity for Intel (12G only)**

This workflow sets IO Identity for Intel cards only. The script is an example for setting IO Identity for the following:

- Port 1
- NIC
- FCoE

**Applies to:** LC2+

**Prerequisites for script:**

- Intel card
- Set desired script variables by editing script
- Intel cards have no partitions

**Script:** IO_IdentityIntel.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus():

B) Check NIC is enabled

1. GetBIOSEnumerations(): ENUMERATE the `DCIM_BIOSEnumeration` class to collect information about the system.
2. Ensure the NIC is enabled
   
   If not enabled, enable it as shown below
   
   - SetBIOSAttributes() Target = (BIOS FQDD)
   - CreateBIOSConfigJob()Target = (BIOS FQDD)
   - ScheduledStartTime=TIME_NOW RebootJobType=1
   - Poll job status for Completed.

C) Disable the FlexAddress

   - SetLCAtributes AttributeName=VirtualAddressManagement AttributeValue=Console

D) Set IO Identity
- SetNICAttributes() Target = (NIC FQDD) AttributeName= VirtMacAddr/ VirtFIPMacAddr/ VirtWWN/VirtWWPN
- CreateNICConfigJob() Target = (NIC FQDD) RebootJobType=1
- Poll job status for completion using instanceID form CreateNICConfigJob()

4.32 Export System Configuration (12th Generation and Later Version of Servers Only)

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

 Applies to: LC2.1+

Prerequisites for script:
- Set script variables by editing script

Script: ExportSystemConfiguration.win

A) The Lifecycle Controller remote service must be in a “ready” state before running any other WSMAN commands. The GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) ExportSystemConfiguration(): Invokes the export operation [ReturnValue=4096]

 NOTE: The user must set applicable IP address, username, password, and filename of the network share.

C) Poll jobstatus for Completed: GET the InstanceID of from B). See section 2.3 for a definition of GET.

Note: With iDRAC 1.50.50+ firmware, by providing a parameter ExportUse in the script, a cloning template with clone and replace options. The details of the cloning feature are available in the Web Services Interface Guide for Windows/Linux (See reference links under Section 1.2)

4.33 Import System Configuration (12th Generation and Later Version of Servers Only)

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

 Applies to: LC2.1+

Prerequisites for script:
- Set script variables by editing script
**Script:** ImportSystemConfiguration.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) ImportSystemConfiguration(): Invokes the import operation [ReturnValue=4096]

**NOTE:** The user must set applicable IP address, username, password, and filename of the network share.

C) Poll jobstatus for Completed: GET the `InstanceID` of from B). See section 2.3 for a definition of GET.

Note: With iDRAC 1.50.50+ firmware, a preview of the errors that will be encountered on a server if a system configuration is imported can be obtained by invoking the ImportSystemConfigurationPreview() function instead of the ImportSystemConfiguration() function.

**4.34 Configurable Boot to Network ISO**

The ConfigurableBootToNetworkISO() method exposes an ISO Image present on a network share as a CDROM device to the host server for a specified exposure duration interval or by default for 18 hrs. Upon the successful execution, based on the ResetType parameter, the host system shall either immediately cold boot or warm boot. Upon this reset, the system shall then boot to the ISO Image. If ResetType specifies no immediate reboot, then upon the next host system reset, the system shall boot to the ISO Image. Furthermore, if immediate reset is not specified, then the system should be rebooted before the exposure duration interval expires, otherwise the system shall fail to boot to the ISO Image.

The workflow first removes any existing driver packs or existing attached OS, then gets available OS drivers, unpacks the desired set of drivers, and boots from an image on a network share. Approximate time for completion on a 12G system is 20-40 minutes depending on the size of the driver pack that will be unpacked and the speed of the network to boot image.

The workflow differs from the 4.3 Boot to Network ISO in that the reboot can be configured for the following:

- Warm boot
- Cold boot
- No reset

**Applies to:** LC2.1+

**Prerequisites for script:**
- Place applicable ISO image in applicable network share
- Set script variables by editing script

**Script:** BootToNetworkISO.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

   **NOTE:** GetRemoteServicesAPIStatus() will return “not ready” if drivers or an ISO is already attached.

B) DetachDrivers(): Ensures any drivers are detached.

C) DetachISOImage(): Ensures all images are detached.

D) GetDriverPackInfo(): Displays available OS drivers. This is only required for end to end OS deployment.

E) UnpackAndAttach(): Unpacks and attaches desired driver pack. The resulting concrete job is invoked immediately. This is only required for end to end OS deployment.

F) Poll concrete job until ‘Success’.

G) BootToNetworkISO(): The resulting concrete job may be invoked immediately depending on the reset type.

H) Poll concrete job until ‘Success’.

5 **Base Metrics Profile Use Cases**

5.1 **Discovery of Base Metrics Profile Support**

Use the following procedure below to confirm the existence of Base Metrics profile support.

**NOTE:** Prior to LC2.0.0, this profile resided as a CIM profile, not LC profile.

**Applies to:** LC1.5.1+

**Prerequisites for script:** none

**Script:** GetBaseMetricsProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) **ENUMERATE** the applicable class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

   [LC1.5.0/LC1.5.1]GetCIMRegisteredProfiles():

   [LC2.0.0]GetLCRegisteredProfiles():

71
C) Search for “RegisteredName= Role Based Authorization” and note its instanceID to use in step D).

D) GET the applicable instance using the InstanceID from C). See Section 2.3 for a definition of GET.

[LC1.5.0/LC1.5.1] GetCIMRegisteredProfile():

[LC2.0.0] GetLCRegisteredProfile():

Results for the InstanceID of DCIM:BaseMetrics:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  AdvertiseTypes = 1
  InstanceID = DCIM:BaseMetrics:1.0.0
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense = Power Monitoring
  ProfileRequireLicenseStatus = LICENSED
  RegisteredName = Base Metrics
  RegisteredOrganization = 1
  RegisteredVersion = 1.0.0
```

6 BIOS and Boot Management Profile Use Cases

6.1 Discovery of BIOS and boot Profile Support

Use the following procedure below to confirm the existence of BIOS and boot profile support.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetBIOSandBootProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSSstatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName= BIOS and Boot Management” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.
Results for the InstanceID of DCIM:BIOSandBootManagement:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  InstanceID = DCIM:BIOSandBootManagement:1.0.0
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense = Remote Firmware Configuration
  ProfileRequireLicenseStatus = LICENSED
  RegisteredName = BIOS and Boot Management
  RegisteredOrganization = 1
  RegisteredVersion = 1.2.0
```

### 6.2 List all BIOS Attributes

Use the following procedure below to view all available instances of the `DCIM_BIOSEnumeration` class, `DCIM_BIOSInteger` class, and `DCIM_BIOSString` classes in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetAllBIOSAttributes.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) **ENUMERATE** the following classes to view all instances. See section 2.2 for a definition of **ENUMERATE**.

- GetBIOSEnumerations(): Returns instance information from DCIM_BIOSEnumeration class
- GetBIOSIntegers(): Returns instance information from DCIM_BIOSInteger class
- GetBIOSStrings(): Returns instance information from DCIM_BIOSString class

The instance information of all available BIOS attributes will be returned.

### 6.3 Delete Pending BIOS Configuration

Use the following procedure below to delete pending BIOS configurations/values set by the setAttribute(s) method.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none
**Script**: DeletePendingBIOSConfiguration.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetBIOSEnumerations (): ENUMERATE the DCIM_BIOSEnumeration class to view all available BIOS FQDDs. See section 2.2 for a definition of ENUMERATE.

C) DeletePendingBIOSConfiguration(): Deletes the pending BIOS configuration, using a target FQDD such as BIOS.Setup.1-1.

A return message of “No pending data present to delete” indicates that there is no pending BIOS configuration to delete for the respective FQDD.

### 6.4 Inventory of boot Configurations in System

Use the following procedure below to view all available instances of the DCIM_BootConfigSetting class.

**Applies to**: LC1.5.0+

**Prerequisites for script**: none

**Script**: GetBootConfigurations.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetBootConfigSettings(): ENUMERATE the DCIM_BootConfigSetting class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available boot configurations will be returned.

### 6.5 Get the First boot Configuration’s Information

Use the following procedure to get a single boot configuration instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

**Applies to**: LC1.5.0+

**Prerequisites for script**:
- Set desired instanceID by editing script (default is IPL)

**Script**: GetBootConfiguration.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
B) GetBootConfigSetting(): GET the DCIM_BootConfigSetting instance using the InstanceID=IPL. See Section 2.3 for a definition of GET.

The instance of DCIM_BootConfigSetting that contains the information on the first boot configuration will be returned.

6.6 Inventory of boot Sources in System
Use the following procedure below to view all available instances of the DCIM_BootSourceSetting class.

 Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetBootSources.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetBootSourceSettings(): ENUMERATE the DCIM_BootSourceSetting class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available boot sources will be returned.

6.7 Changing boot Order by Instance
See Section “4.5: Set hard drive to first in boot order” for a comprehensive example.

6.8 Enable or Disable boot Source
This workflow can enable[1] (or disable[0]) the hard drive(c:). If the BootMode is set to UEFI, it will change it to BIOS. Following completion of the reboot, the hard drive will be enabled (or disabled). Approximate time for completion that one can expect to encounter on an 11G system is about 15-25 minutes, depending on whether an additional reboot is needed to set the BootMode parameter.

 Applies to: LC1.5.0+

Prerequisites for script:

- Set the desired boot source state to enable(1) or disable(0)

Script: EnableBootSource.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) Change BootMode to BIOS, if current value is UEFI

   a. SetAttribute(): Sets attribute to be configured [ReturnValue=0]
b. **CreateConfigJob()**: Creates jobID and applies configuration [ReturnVal=4096]

C) **GetBootConfigSettings()**: `ENUMERATE` the `DCIM_BootConfigSetting` class to identify the `ElementName` field containing `BootSeq` and corresponding `InstanceID` (IPL or UEFI). See section 2.2 for a definition of `ENUMERATE`.

   `ElementName = Hard drive C: BootSeq`

D) **GetBootSourceSettings()**: `ENUMERATE` the `DCIM_BootSourceSetting` class. See section 2.2 for a definition of `ENUMERATE`.

   a. The `CurrentEnabledStatus` attribute defines whether the boot source is enabled or disabled

   b. If the `CurrentEnabledStatus` is desired value, skip to the end

E) **ChangeBootSourceState()**: using `instanceID = IPL` and `EnabledState=1` [ReturnVal=0]

F) **Poll jobstatus for Completed**: `GET` the `InstanceID` of from E). See section 2.3 for a definition of `GET`.

G) **ENUMERATE** the `DCIM_BootSourceSetting` class. See section 2.2 for a definition of `ENUMERATE`.

   - The `CurrentEnabledStatus` of the “Hard drive C” should be 1 for enable (or 0 for disable)

6.9 **One Time boot**

This workflow sets a one-time boot for vFlash, IPL, or UEFI. This example uses IPL. Approximate time for completion that one can expect to encounter is about 15-25 minutes.

**Applies to**: LC1.5.0+

**Prerequisites for script**:

- none

**Script**: OneTimeBoot.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

   The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

G) **ENUMERATE** the `DCIM_BootSourceSetting` class. See section 2.2 for a definition of `ENUMERATE`.

   - The `CurrentEnabledStatus` of the “Hard drive C” should be 1 for enable (or 0 for disable)
C) GetBootSourceSettings(): ENUMERATE the `DCIM_BootSourceSetting` class and identify the desired one time boot source: vFlash, IPL, or UEFI

D) ChangeBootOrderByInstanceID(): using instanceID = OneTime and Source=instanceID from C)

E) GetBootConfigSettings(): ENUMERATE the `DCIM_BootConfigSetting` class.
   - The `OneTime` entry should have an `IsNext` value of 3, which means “Is Next for Single Use”

F) GetBootSourceSettings(): ENUMERATE the `DCIM_BootSourceSetting` class.
   - Verify that an additional entry for that `DCIM_BootSourceSetting` appears with an `InstanceID` prefixed with “OneTime:”, such as `InstanceID = OneTime:IPL:HardDisk.List.1-1:c9203080df84781e2ca3d512883dee6f`.
   - After the reboot, the boot list reverts to the original boot list.

The job will be marked ‘Completed’ after the successful one time boot to the device.

7 CPU Profile Use Cases

7.1 Discovery of CPU Profile Support
Use the following procedure below to confirm the existence of CPU profile support.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetCPUProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the `DCIM_LCRegisteredProfile` class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=CPU” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the `DCIM_LCRegisteredProfile` instance using the `InstanceID` from C). See Section 2.3 for a definition of GET.

Results for the `InstanceID` of DCIM:CPU:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
AdvertiseTypeDescriptions = WS-Identify, Interop Namespace
AdvertiseTypes = 1, 1
InstanceID = DCIM:CPU:1.0.0
OtherRegisteredOrganization = DCIM
ProfileRequireLicense = Remote Inventory
```
7.2 Inventory of CPUs in System

Use the following procedure below to list the inventory of all CPUs in the system.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetCPUViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetCPUViews(): ENUMERATE the DCIM_CPUView class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available CPUs will be returned.

7.3 Get the First CPU’s Information

Use the following procedure to get a single CPU instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

Applies to: LC1.5.0+

Prerequisites for script:

- Set desired instanceID by editing script

Script: GetCPUView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetCPUView(): GET the DCIM_CPUView instance using the InstanceID=CPU.Socket.1. See Section 2.3 for a definition of GET.

The instance of DCIM_CPUView that contains the information on the first CPU will be returned.

8 Event Filter Profile Use Cases

8.1 Discovery of Event Filter Profile Support

Use the following procedure below to confirm the existence of Event Filter profile support. This profile is currently not supported prior to LC2.
Applies to: LC2+

Prerequisites for script: none

Script: GetEventFilterProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

    GetRemoteServicesAPIStatus():

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Event Filter” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:EventFilter:1.0.0 shown below. If no instance is returned, the profile is not supported.

    DCIM_LCRegisteredProfile
    AdvertiseTypeDescriptions = WS-Identify
    AdvertiseTypeDescriptions = Interop Namespace
    AdvertiseTypes = 1
    AdvertiseTypes = 1
    InstanceID = DCIM:EventFilter:1.0.0
    OtherRegisteredOrganization = DCIM
    ProfileRequireLicense = Remote Firmware Configuration
    ProfileRequireLicenseStatus = LICENSED
    RegisteredName = Event Filter
    RegisteredOrganization = 1
    RegisteredVersion = 1.0.0

8.2 Get Event Filter Configuration Service Views

Enumerate the DCIM_EFConfigurationService class to view all available instances of the class.

Applies to: LC2+

Prerequisites for script: none

Script: GetEFConfigurationServiceViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

    GetRemoteServicesAPIStatus():

B) GetEFConfigurationServiceViews(): ENUMERATE the DCIM_EFConfigurationService class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information will be returned.
8.3 Get Event Filter Views
Enumerate the DCIM_EventFilter class to view all available instances of the class.

**Applies to:** LC2+

**Prerequisites for script:** none

**Script:** GetEventFilterViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   - GetRemoteServicesAPIStatus():

B) GetEventFilters(): ENUMERATE the DCIM_EventFilter class to view all instances. See section 2.2 for a definition of ENUMERATE.

All available instance information will be returned.

8.4 Get Single Event Filter’s Information
Use the following procedure to get a single Event Filter instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

**Applies to:** LC2+

**Prerequisites for script:**
- Set desired instanceID by editing script

**Script:** GetEventFilterView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   - GetRemoteServicesAPIStatus():

B) GetEventFilterView(): GET the DCIM_EventFilter instance. See Section 2.3 for a definition of GET.

The desired instance information will be returned.

8.5 Set Event Filters by Category
This workflow is used to set the action and notifications for all the event filters that belong to a particular category, subcategory and severity.

**Applies to:** LC2+

**Prerequisites for script:** Set variables by editing script

**Script:** SetEventFilterByCategory.win
A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus():

B) SetEventFilterByCategory(): Sets the action and notifications for the event filters

8.6 Set Event Filters by InstanceID
This workflow is used to set the action and notifications for all the event filters that belong to a particular set of InstanceIDs.

Applies to: LC2+

Prerequisites for script: Set variables by editing script

Script: SetEventFilterByInstanceIDs.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus():

B) GetEventFilters(): ENUMERATE the DCIM_EventFilter class to view all available instances to use in C). See section 2.2 for a definition of ENUMERATE.

C) SetEventFilterByInstanceIDs(): Sets the action and notifications for the event filters

9 iDRAC Card Profile Use Cases

9.1 Discovery of iDRAC Card Profile Support
Use the following procedure below to confirm the existence of iDRAC Card profile support.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetiDRACCardProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=iDRAC Card” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.
Results for the InstanceID of DCIM:iDRACCard:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
AdvertiseTypeDescriptions = WS-Identify
AdvertiseTypeDescriptions = Interop Namespace
AdvertiseTypes = 1
InstanceID = DCIM:iDRACCard:1.0.0
ProfileRequireLicense = Remote Firmware Configuration
ProfileRequireLicense = Remote Inventory
ProfileRequireLicense = Virtual Console
ProfileRequireLicense = Virtual Media
ProfileRequireLicense = Two-Factor Authentication
ProfileRequireLicense = Directory Services
ProfileRequireLicense = IPv6
ProfileRequireLicense = Dynamic DNS
ProfileRequireLicenseStatus = LICENSED
ProfileRequireLicenseStatus = LICENSED
ProfileRequireLicenseStatus = LICENSED
ProfileRequireLicenseStatus = LICENSED
ProfileRequireLicenseStatus = LICENSED
ProfileRequireLicenseStatus = LICENSED
RegisteredName = iDRAC Card
RegisteredOrganization = 1
RegisteredVersion = 1.2.0
```

9.2 **Get all iDRAC Card Attributes**

Use the following procedure below to list all of the attributes from all of the iDRAC Card class in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetAlliDRACCardAttributes.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetiDRACCardAttributes(): ENUMERATE the DCIM_iDRACCardAttribute class to view all attributes. See section 2.2 for a definition of ENUMERATE.

9.3 **Inventory of iDRAC Cards in System**

Use the following procedure below to list the inventory of all iDRAC Cards in the system.
Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetiDRACCardViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetiDRACCardViews(): ENUMERATE the DCIM_iDRACCardView class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available iDRAC cards will be returned.

9.4 Get the First iDRAC Card’s Information

Use the following procedure to get a single iDRAC instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

Applies to: LC1.5.0+

Prerequisites for script:

- Set desired InstanceID by editing script

Script: GetiDRACCardView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetiDRACCardView(): GET the DCIM_iDRACCardView instance using the InstanceID= iDRAC.Embedded.1. See Section 2.3 for a definition of GET.

The instance of DCIM_iDRACCardView that contains the information on the first iDRAC card will be returned.

9.5 Set Apply iDRAC Card Attribute(s) Immediately

Use the following procedure to set a iDRAC attributes(s) immediately using the ApplyAttributes method. It is not necessary to invoke the CreateTargetConfigJob method as with the SetAttributes method.

Applies to: LC1.5.0+

Prerequisites for script: Set desired values by editing script

Script: ApplyiDRACCardAttributes.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetiDRACCardAttributes(): ENUMERATE the DCIM_iDRACCardAttribute class to view all available attributes. See section 2.2 for a definition of ENUMERATE.
C) Confirm the IsReadOnly field is set to false
D) ApplyAttributes(): Invoke method to apply attribute(s)
E) GetiDRACCardAttributes(): ENUMERATE the DCIM_iDRACCardAttribute class to view all available attributes and confirm the changes were successful. See section 2.2 for a definition of ENUMERATE.

9.6 Schedule a set iDRAC Card Attribute(s) Operation
Use the following procedure to set a iDRAC attribute(s) using the SetAttributes and CreateTargetConfigJob methods. This example sets the attribute immediately using the TIME_NOW parameter, however the job can be scheduled for execution at a later time.

Applies to: LC2.0+

Prerequisites for script: Set desired values by editing script

Script: SetiDRACCardAttribute.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus():
B) GetiDRACCardAttributes(): ENUMERATE the DCIM_iDRACCardAttribute class to view all available attributes. See section 2.2 for a definition of ENUMERATE.
C) Confirm the IsReadOnly field is set to false for desired instances
D) SetAttributes(): Invoke method [ReturnValue = 0]
E) CreateiDRACConfigJob(): Apply pending values using TIME_NOW parameter
F) GetiDRACCardAttributes(): ENUMERATE the DCIM_iDRACCardAttribute class to view all available attributes and confirm the changes were successful. See section 2.2 for a definition of ENUMERATE.

10 Fan Profile Use Cases

10.1 Discovery of Fan Profile Support
Use the following procedure below to confirm the existence of fan profile support.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetFanProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
B) GetLCRegisteredProfiles(): ENUMERATE the \texttt{DCIM\_LCRegisteredProfile} class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Fan” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the \texttt{DCIM\_LCRegisteredProfile} instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:Fan:1.0.0 shown below. If no instance is returned, the profile is not supported.

\begin{verbatim}
DCIM\_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify, Interop Namespace
  AdvertiseTypes = 1, 1
  InstanceID = DCIM:Fan:1.0.0
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense = Remote Inventory
  ProfileRequireLicenseStatus = LICENSED
  RegisteredName = Fan
  RegisteredOrganization = 1
  RegisteredVersion = 1.0.0
\end{verbatim}

10.2 Inventory of Fans in System

Use the following procedure below to list the inventory of all fans in the system.

\textbf{Applies to:} LC1.5.0+

\textbf{Prerequisites for script:} none

\textbf{Script:} GetFanView\_\_win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetFanViews(): ENUMERATE the \texttt{DCIM\_FanView} class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available fans will be returned.

10.3 Get the First Fan’s Information

Use the following procedure to get a single fan instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

\textbf{Applies to:} LC1.5.0+

\textbf{Prerequisites for script:}

- Set desired instanceID by editing script

\textbf{Script:} GetFanView\_\_win
A) The Lifecycle Controller remote service must be in a “ready” state before executing any 
other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() 
method may be used depending on the version of the LC Management registered profile.

B) GetCPUView(): GET the DCIM_FanView instance using the InstanceID=Fan.embedded.1. 
See Section 2.3 for a definition of GET.

The instance of DCIM_FanView that contains the information on the first fan will be returned.

11 Persistent Storage Profile Use Cases

11.1 Discovery of Persistent Storage Profile Support

Use the following procedure below to confirm the existence of Persistent Storage profile support.

**Applies to:** LC1.4.0+

**Prerequisites for script:** none

**Script:** GetPersistentStorageProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any 
other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() 
method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all 
registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Persistent Storage” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the 
InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:PersistentStorage:1.0.0 shown below. If no instance is 
returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  AdvertiseTypes = 1
  InstanceID = DCIM:PersistentStorage:1.0.0
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense = Virtual Flash Partitions
  ProfileRequireLicense = Remote Inventory
  ProfileRequireLicenseStatus = LICENSED
  ProfileRequireLicenseStatus = LICENSED
  RegisteredName = Persistent Storage
  RegisteredOrganization = 1
  RegisteredVersion = 1.0.0
```

11.2 Inventory of Virtual Flash (vFlash) Media

Use the following procedure below to list the inventory of all vFlash in the system.
11.3 Get the First vFlash’s Attribute Information

The URI for getting particular instance information is deterministic (i.e., the InstanceID will be unique for each instance)

**Applies to:** LC1.5.0+

**Prerequisites for script:**
- Set desired InstanceID by editing script

**Script:** GetVFlashView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.

B) GetVFlashView(): GET the `DCIM_VFlashView` instance using the InstanceID=`Disk.vFlashCard.1`. See Section 2.3 for a definition of GET.

The instance of `DCIM_VFlashView` that contains the information on the first vFlash will be returned.

11.4 Inventory of Partitions on the Virtual Flash Media

Use the following procedure below to list the inventory of all vFlash partitions in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetVFlashPartitionViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetVFlashPartitionViews(): ENUMERATE the `DCIM_OpaqueManagementData` class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available partitions will be returned.
11.5 **Initialize Virtual Flash Media**
This method is used to initialize or format the virtual flash media device.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** InitVFlash.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetVFlashViews(): **ENUMERATE** the **DCIM_VFlashView** class to view the current value of InitializedState. See section 2.2 for a definition of **ENUMERATE**.

C) InitializeMedia(): Invokes the InitializeMedia method on the class **DCIM_PersistentStorageService**

D) Poll jobstatus for Completed: **GET** the **InstanceID** of from C). See section 2.3 for a definition of **GET**.

E) GetVFlashViews(): **ENUMERATE** the **DCIM_VFlashView** class to confirm the new value of InitializedState. See section 2.2 for a definition of **ENUMERATE**.

11.6 **Enable Virtual Flash (vFlash) Media**
This method is used to enable the virtual flash media device.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** EnableVFlash.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetVFlashViews(): **ENUMERATE** the **DCIM_VFlashView** class to view the current value of VFlashEnabledState property. See section 2.2 for a definition of **ENUMERATE**.

C) VFlashStateChange(): Invokes the VFlashStateChange method on the class **DCIM_PersistentStorageService**

D) Repeat B) to confirm successful execution of the method

11.7 **Disable Virtual Flash (vFlash) Media**
This method is used to disable the virtual flash media device.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none
Script: DisableVFlash.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetVFlashViews(): ENUMERATE the DCIM_VFlashView class to view the current value of VFlashEnabledState property. See section 2.2 for a definition of ENUMERATE.

C) VFlashStateChange(): Invokes the VFlashStateChange method on the class DCIM_PersistentStorageService

D) Repeat B) to confirm successful execution of the method

11.8 Create new Partition on Virtual Flash (vFlash) Media

This method is used to create a new partition on the virtual flash media device.

Applies to: LC1.5.0+

Prerequisites for script: set variables by editing script

Script: VFlashCreatePartition.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetVFlashPartitionViews(): ENUMERATE the DCIM_OpaqueManagementData class to view the current partitions. See section 2.2 for a definition of ENUMERATE.

C) VFlashStateChange(): Enable vFlash if it is disabled

D) CreatePartition(): Invokes the CreatePartition method on the class DCIM_PersistentStorageService

E) Poll jobstatus for Completed: GET the InstanceID of from E). See section 2.3 for a definition of GET.

F) Repeat B) to confirm successful execution of the method

11.9 Create new Partition Using Image

This method is used to create a new partition on the virtual flash media device using an image from a network share.

Applies to: LC1.5.0+

Prerequisites for script: set variables by editing script

Script: VFlashCreatePartitionUsingImage.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
B) GetVFlashPartitionViews(): \textsc{enumerate} the \textit{DCIM\textunderscore\textunderscore OpaqueManagementData} class to view the current partitions. See section 2.2 for a definition of \textsc{enumerate}.

C) CreatePartitionUsingImage(): Invokes the CreatePartitionUsingImage method on the class \textit{DCIM\textunderscore\textunderscore PersistentStorageService}

D) Poll jobstatus for Completed: \textsc{get} the \textit{InstanceId} of from C). See section 2.3 for a definition of \textsc{get}.

E) Repeat B) to confirm successful execution of the method

11.10 Delete Existing Partition
This method is used to delete a partition from the virtual flash media device.

\textbf{Applies to:} LC1.5.0+

\textbf{Prerequisites for script:} set variables by editing script

\textbf{Script:} VFlashDeletePartition.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The \textit{GetRSStatus()} method or the \textit{GetRemoteServicesAPIStatus()} method may be used depending on the version of the LC Management registered profile.

B) GetVFlashPartitionViews(): \textsc{enumerate} the \textit{DCIM\textunderscore\textunderscore OpaqueManagementData} class to view the current partitions. See section 2.2 for a definition of \textsc{enumerate}.

C) DeletePartition(): Invokes the \textit{DeletePartition} method on the class \textit{DCIM\textunderscore\textunderscore PersistentStorageService}

D) GetVFlashPartitionViews(): \textsc{enumerate} the \textit{DCIM\textunderscore\textunderscore OpaqueManagementData} class to view the current partitions and confirm successful operation. See section 2.2 for a definition of \textsc{enumerate}.

11.11 Format Existing Partition
This method is used to format a partition on the virtual flash media device.

\textbf{Applies to:} LC1.5.0+

\textbf{Prerequisites for script:} set variables by editing script

\textbf{Script:} VFlashFormatPartition.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The \textit{GetRSStatus()} method or the \textit{GetRemoteServicesAPIStatus()} method may be used depending on the version of the LC Management registered profile.

B) GetVFlashViews(): \textsc{enumerate} the \textit{DCIM\textunderscore\textunderscore VFlashView} class to view the current partitions, enable vFlash if disabled. See section 2.2 for a definition of \textsc{enumerate}.

C) FormatPartition(): Invokes the \textit{FormatPartition} method on the class \textit{DCIM\textunderscore\textunderscore PersistentStorageService}
D) Poll jobstatus for Completed: GET the InstanceID of from C). See section 2.3 for a definition of GET.

E) GetVFlashPartitionViews(): ENUMERATE the DCIM_OpaqueManagementData class to view the current partitions and confirm successful operation. See section 2.2 for a definition of ENUMERATE.

### 11.12 Modify Existing Partition

This method is used to modify a partition on the virtual flash media device.

**Applies to:** LC1.5.0+

**Prerequisites for script:** set variables by editing script

**Script:** VFlashModifyPartition.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetVFlashViews(): ENUMERATE the DCIM_VFlashView class to view the current partitions. See section 2.2 for a definition of ENUMERATE.

C) ModifyPartition(): Invokes the ModifyPartition method on the class DCIM_PersistentStorageService

D) GetVFlashPartitionViews(): ENUMERATE the DCIM_OpaqueManagementData class to view the current partitions and confirm successful operation. See section 2.2 for a definition of ENUMERATE.

### 11.13 Attach Partition

This method is used to attach a partition on the virtual flash media device.

**Applies to:** LC1.5.0+

**Prerequisites for script:** set variables by editing script

**Script:** VFlashAttachPartition.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) AttachPartition(): Invokes the AttachPartition method on the class DCIM_PersistentStorageService

C) Poll jobstatus for Completed: GET the InstanceID of from B). See section 2.3 for a definition of GET.
D) GetVFlashPartitionViews(): ENUMERATE the DCIM_OpaqueManagementData class to view the current partitions and confirm successful operation. See section 2.2 for a definition of ENUMERATE.

11.14 Detach Partition
This method is used to detach a partition on the virtual flash media device.

 Applies to: LC1.5.0+

Prerequisites for script: set variables by editing script

Script: VFlashDetachPartition.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) DetachPartition(): Invokes the DetachPartition method on the class DCIM_PersistentStorageService

C) Poll jobstatus for Completed: GET the InstanceID of from B). See section 2.3 for a definition of GET.

D) GetVFlashPartitionViews(): ENUMERATE the DCIM_OpaqueManagementData class to view the current partitions and confirm successful operation. See section 2.2 for a definition of ENUMERATE.

11.15 Export Data from Existing Partition
This method is used to export data from a partition on the virtual flash media device to a network share.

 Applies to: LC1.5.0+

Prerequisites for script: set variables by editing script

Script: VFlashExportDataFromPartition.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) ExportDataFromPartition(): Invokes the ExportDataFromPartition method on the class DCIM_PersistentStorageService

C) Poll jobstatus for Completed: GET the InstanceID of from B). See section 2.3 for a definition of GET.

D) GetVFlashPartitionViews(): ENUMERATE the DCIM_OpaqueManagementData class to view the current partitions and confirm successful operation. See section 2.2 for a definition of ENUMERATE.
12 Power State Management Profile Use Cases

12.1 Discovery of Power State Management Profile Support

Use the following procedure below to confirm the existence of Power State Management profile support. NOTE: Prior to LC2.0.0, this profile resided as a CIM profile, not LC profile.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetPowerStateMGMTProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) [LC1.5.0/LC1.5.1] GetCIMRegisteredProfiles():
   [LC2.0.0] GetLCRegisteredProfiles():
   ENUMERATE the applicable class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Power State Management” and note its instanceID to use in step D)

D) [LC1.5.0/LC1.5.1] GetCIMRegisteredProfile():
   [LC2.0.0] GetLCRegisteredProfile():
   GET the applicable instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:PowerStateManagement:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
 AdvertiseTypeDescriptions = WS-Identify
 AdvertiseTypeDescriptions = Interop Namespace
 AdvertiseTypes = 1
 AdvertiseTypes = 1
 InstanceID = DCIM:PowerStateManagement:1.0.0
 OtherRegisteredOrganization = DCIM
 ProfileRequireLicense
 ProfileRequireLicenseStatus
 RegisteredName = Power State Management
 RegisteredOrganization = 1
 RegisteredVersion = 1.0.0
```
13 Profile Registration Profile Use Cases

13.1 Discovery of Profile Registration Profile Support
Use the following procedure below to confirm the existence of profile registration profile support.

**Applies to:** LC2+

**Prerequisites for script:** none

**Script:** GetProfileRegistrationProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   GetRemoteServicesAPIStatus();

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Profile Registration” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:ProfileRegistrationProfile shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  AdvertiseTypes = 1
  InstanceID = DCIM:ProfileRegistrationProfile
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense
  ProfileRequireLicenseStatus
  RegisteredName = Profile Registration
  RegisteredOrganization = 1
  RegisteredVersion = 1.0.0
```

14 Simple RAID Profile Use Cases

14.1 Discovery of RAID Profile Support

Use the following procedure below to confirm the existence of RAID profile support.

**Applies to:** LC1.5.0+
Prerequisites for script: none

Script: GetRAIDProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Simple RAID” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:SimpleRAID:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
AdvertiseTypeDescriptions = WS-Identify
AdvertiseTypeDescriptions = Interop Namespace
AdvertiseTypes = 1
AdvertiseTypes = 1
InstanceID = DCIM:SimpleRAID:1.0.0
OtherRegisteredOrganization = DCIM
ProfileRequireLicense = Remote Firmware Configuration
ProfileRequireLicense = Remote Inventory
ProfileRequireLicenseStatus = LICENSED
ProfileRequireLicenseStatus = LICENSED
RegisteredName = Simple RAID
RegisteredOrganization = 1
RegisteredVersion = 1.2.0
```

14.2 Inventory of RAID Controllers in System

Use the following procedure below to list the inventory of all RAID controllers in the system.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetRAIDControllerViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetControllerViews(): ENUMERATE the DCIM_ControllerView class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available RAID controllers will be returned
14.3 Get the first RAID Controller’s Information

Use the following procedure to get a single RAID controller instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

**Applies to:** LC1.5.0+

**Prerequisites for script:**

- Set desired instanceID by editing script

**Script:** GetRAIDControllerView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIServiceStatus() method may be used depending on the version of the LC Management registered profile.

B) GetControllerView(): **GET** the DCIM_ControllerView instance using the desired instanceID. See Section 2.3 for a definition of **GET**.

The instance of DCIM_ControllerView that contains the information on the first RAID controller will be returned.

14.4 Inventory of Virtual and Physical Disk Drives in System

Use the following procedure below to list the inventory of all virtual disks and physical disks in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetRAIDDiskInventory.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIServiceStatus() method may be used depending on the version of the LC Management registered profile.

B) GetVirtualDiskViews(): **ENUMERATE** the DCIM_VirtualDiskView class to view all instances. See section 2.2 for a definition of **ENUMERATE**.

C) GetPhysicalDiskViews(): **ENUMERATE** the DCIM_PhysicalDiskView class to view all instances. See section 2.2 for a definition of **ENUMERATE**.

The instance information of all available virtual disks and physical disks will be returned.

14.5 Apply Pending Values for a RAID Configuration

View the CreateRAIDConfigJob() step in the RAID stacking workflows in Section 4 for a comprehensive example.

14.6 Delete Pending Values for a RAID Configuration

Use the following procedure below to delete pending RAID configurations/values method.

**Applies to:** LC1.5.0+
Prerequisites for script:

- Set desired RAID FQDD by editing script

Script: DeletePendingRAIDConfiguration.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetControllerViews(): ENUMERATE the DCIM_ControllerView class to view all available RAID FQDDs. See section 2.2 for a definition of ENUMERATE.

C) DeletePendingRAIDConfiguration(): Deletes the pending configuration for a particular RAID controller, using a target FQDD.

14.7 Clear old Configuration from Newly Added HDD

Use the following procedure below to clear old configuration from newly added hard drives.

Applies to: LC1.5.0+

Prerequisites for script:

- Set desired RAID FQDD by editing script

Script: ClearForeignConfig.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetControllerViews(): ENUMERATE the DCIM_ControllerView class to view all available RAID FQDDs. See section 2.2 for a definition of ENUMERATE.

C) ClearForeignConfig(): Clears the configuration for a particular RAID controller, using a target FQDD.

D) CreateRAIDConfigJob(): Apply step C) [ReturnValue=4096].

E) Poll jobstatus for Completed: GET the InstanceID of from D). See section 2.3 for a definition of GET.

F) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

14.8 Determine Available RAID Configurations

Use the following procedure below to list the available RAID level configurations for a given set of physical disks.
Applies to: LC1.5.0+

Prerequisites for script:

- Set desired RAID FQDD by editing script

Script: GetRAIDLevels.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetControllerViews(): ENUMERATE the DCIM_ControllerView class to view all available RAID FQDDs. See section 2.2 for a definition of ENUMERATE.

C) GetRAIDLevels(): Invoke this method to return desired data
   The output will contain the available RAID level configurations for the given physical disk selection

14.9 Determine Available Physical Disk Drives for a RAID Configuration

Use the following procedure below to list the available physical disks for a given RAID level configurations.

Applies to: LC1.5.0+

Prerequisites for script:

- Set desired RAID FQDD by editing script

Script: GetRAIDAvailableDisks.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetControllerViews(): ENUMERATE the DCIM_ControllerView class to view all available RAID FQDDs. See section 2.2 for a definition of ENUMERATE.

C) GetAvailableDisks(): Invoke this method which will return the desired data
   The output will contain the available physical disks for the given RAID level.

14.10 Check Available Virtual Disk Parameters for a given RAID Level and set of Physical Disks

Use the following procedure below to list the available sizes and default values for a given RAID level configurations.

Applies to: LC1.5.0+

Prerequisites for script:
- Set desired RAID controller FQDD, physical disk FQDDs, and RAID level parameters by editing script

**Script:** CheckVDValues.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetControllerViews(): **ENUMERATE** the `DCIM_ControllerView` class to view all available RAID FQDDs. See section 2.2 for a definition of **ENUMERATE**.

C) GetPhysicalDiskViews(): **ENUMERATE** the `DCIM_PhysicalDiskView` class to view all available physical disk FQDDs. See section 2.2 for a definition of **ENUMERATE**.

D) CheckVDValues(): Invoke this method to return desired data

The output will contain the available sizes and default values for the given RAID level and set of physical disks

### 14.11 Create a Virtual Disk

View the RAID stacking workflows in Section 4 for a comprehensive example.

### 14.12 Determine Available Physical Disk Drives to be used as a Hot-spare

The GetDHSDisks() method is used to determine possible physical disks to be used as a hotspare.

**Applies to:** LC1.5.0+

**Prerequisites for script:**

- Set desired RAID controller FQDD, physical disk FQDDs, and RAID level parameters by editing script

**Script:** GetDHSDisks.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetControllerViews(): **ENUMERATE** the `DCIM_ControllerView` class to view all available RAID FQDDs. See section 2.2 for a definition of **ENUMERATE**.

C) GetDHSDisks(): Invoke this method to return desired data

The output will contain the available physical disks for use as a hotspare.

### 14.13 Assign a Physical Disk Drive as a Hot-spare

View the RAID stacking workflows in Section 4 for a comprehensive example.

### 14.14 Delete a Virtual Disk from the System
The following script can be used to delete an existing virtual disk from a system.

**Applies to:** LC1.5.0+

**Prerequisites for script:**
- Set desired RAID virtual disk FQDD by editing script

**Script:** DeleteVirtualDisk.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetControllerViews(): **ENUMERATE** the DCIM_ControllerView class to view all available RAID FQDDs. See section 2.2 for a definition of **ENUMERATE**.

C) GetVirtualDiskViews(): **ENUMERATE** the DCIM_VirtualDiskView class to view all available virtual disk FQDDs. See section 2.2 for a definition of **ENUMERATE**.

D) DeleteVirtualDisk(): Invoke this method to delete the VD

E) CreateRAIDConfigJob(): Apply step D) [ReturnValue=4096]

F) Poll jobstatus for Completed: **GET** the InstanceID of from E). See section 2.3 for a definition of **GET**.

G) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

   The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPIStatus() can just poll for ‘ready.’

H) **ENUMERATE** the DCIM_VirtualDiskView class to ensure successful virtual disk deletion. See section 2.2 for a definition of **ENUMERATE**.

   Confirm successful execution of the method by listing the virtual disks. The deleted virtual disk should not be displayed.

### 14.15 Delete all Virtual Disks and Unassign all Hot-sares

View the RAID stacking workflows in Section 4 for a comprehensive example.

### 14.16 Convert Physical Disk Drive to RAID State

The ConvertToRAID() method is used to convert a 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
DCIM_PhysicalDiskView.RAIDStatus property of that physical disk should reflect the new state. One can expect this operation to take up to 15 minutes depending on system configuration.

**Applies to:** LC2+

**Prerequisites for script:**

- Set desired RAID physical disk FQDD by editing script

**Script:** ConvertToRAID.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

   GetRemoteServicesAPIStatus()

B) GetControllerViews(): ENUMERATE the DCIM_ControllerView class to view all available RAID FQDDs. See section 2.2 for a definition of ENUMERATE.

C) GetPhysicalDiskViews(): ENUMERATE the DCIM_PhysicalDiskView class to view all available physical disk FQDDs. See section 2.2 for a definition of ENUMERATE.

Note the value of the RaidStatus parameter of the desired physical disk.

D) ConvertToRAID(): Invoke this method

E) CreateRAIDConfigJob(): Apply step D) [ReturnValue=4096]

F) Poll jobstatus for Completed: GET the InstanceID of from E). See section 2.3 for a definition of GET.

G) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

   GetRemoteServicesAPIStatus()

H) GetPhysicalDiskViews(): ENUMERATE the DCIM_PhysicalDiskView class to view all available physical disk FQDDs. See section 2.2 for a definition of ENUMERATE.

Note the new value of the RaidStatus parameter of the desired physical disk.

14.17 Convert Physical Disk Drives to non-RAID State

The ConvertToNonRAID() method is used to convert a set of physical disks in RAID state to a state not 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 DCIM_PhysicalDiskView.RAIDStatus property of that physical disk should reflect the new state. One can expect this operation to take up to 15 minutes depending on system configuration.

**Applies to:** LC2+

**Prerequisites for script:**
• Set desired RAID physical disk FQDD by editing script

**Script:** ConvertToNonRAID.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus()

B) `GetControllerViews()`: `ENUMERATE` the `DCIM_ControllerView` class to view all available RAID FQDDs. See section 2.2 for a definition of `ENUMERATE`.

C) `GetPhysicalDiskViews()`: `ENUMERATE` the `DCIM_PhysicalDiskView` class to view all available physical disk FQDDs. See section 2.2 for a definition of `ENUMERATE`.

Note the value of the `RaidStatus` parameter of the desired physical disk.

D) `ConvertToNonRAID()`: Invoke this method

E) `CreateRAIDConfigJob()`: Apply step D) [ReturnValue=4096]

F) Poll jobstatus for Completed: `GET` the `InstanceId` of from E). See section 2.3 for a definition of `GET`.

G) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus()

H) `GetPhysicalDiskViews()`: `ENUMERATE` the `DCIM_PhysicalDiskView` class to view all available physical disk FQDDs. See section 2.2 for a definition of `ENUMERATE`.

Note the new value of the `RaidStatus` parameter of the desired physical disk.

### 15 Record Log Profile Use Cases

#### 15.1 Discovery of Record Log Profile Support

Use the following procedure below to confirm the existence of Record Log profile support.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetRecordLogProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The `GetRSStatus()` method or the `GetRemoteServicesAPIStatus()` method may be used depending on the version of the LC Management registered profile.
B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Record log” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:RecordLog:1.0.0 shown below. If no instance is returned, the profile is not supported.

```plaintext
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  AdvertiseTypes = 1
  InstanceID = DCIM:RecordLog:1.0.0
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense
  ProfileRequireLicenseStatus
  RegisteredName = Record Log
  RegisteredOrganization = 1
  RegisteredVersion = 1.0.0
```

15.2 List Lifecycle Record Logs

Use the following procedure below to list the inventory of all LCRecordLog instances in the system.

- **Applies to:** LC1.5.0+
- **Prerequisites for script:** none
- **Script:** GetLCRecordLogs.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRecordLogs(): ENUMERATE the DCIM_LCRecordLog class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available LCRecordLog will be returned.

15.3 List Lifecycle Record Log Capabilities

Use the following procedure below to list the inventory of all LCRecordLogCapabilities instances in the system.

- **Applies to:** LC1.5.0+
- **Prerequisites for script:** none
**Script:** GetLCRecordLogCapabilities.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRecordLogCapabilities(): ENUMERATE the DCIM_LCRecordLogCapabilities class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available LCRecordLogCapabilities will be returned.

### 15.4 List Lifecycle Log Entries

Use the following procedure below to list the inventory of all LCLogEntry instances in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:**

- Due to the large amount of data returned by this enumeration, winRM settings may need to be changed as described in the section 1.2

**Script:** GetLCLogEntries.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCLogEntries(): ENUMERATE the DCIM_LCLogEntry class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available LCLogEntry will be returned.

### 15.5 Set and get Comment in Lifecycle Log Entries

Use the following procedure below to ‘Get’ and ‘Set’ the instance of DCIM_LCLogEntry with the Comment that needs to be added. This is accomplished using an intrinsic set/put operation on the DCIM_LCLogEntry instance at the Comment property.

**Applies to:** LC1.5.0+

**Prerequisites for script:**

- Due to the large amount of data returned by this enumeration, winRM settings may need to be changed as described in the section 1.2
- Set desired instanceID by editing script

**Script:** SetLCLogEntryComment.win
A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCLogEntries(): ENUMERATE the DCIM_LCLogEntry class to view all instances of the class LCLogEntry. See section 2.2 for a definition of ENUMERATE.

C) GetLCLogEntry(): GET the desired InstanceID of from B). See section 2.3 for a definition of GET. This will display the current comment.

D) SetLCLogEntryComment(): Set the desired comment into the InstanceID of from C)

E) GetLCLogEntry(): GET the desired InstanceID of from B) to confirm new comment

15.6 List System Event Record Logs
Use the following procedure below to list the inventory of all SELRecordLog instances in the system.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetSystemEventLogs.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSystemEventLogs(): ENUMERATE the DCIM_SELRecordLog class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available SELRecordLog will be returned

15.7 List System Event Record Log Capabilities
Use the following procedure below to list the inventory of all SELRecordLogCapabilities instances in the system.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetSystemEventLogCapabilities.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSystemEventLogCapabilities(): ENUMERATE the DCIM_SELRecordLogCapabilities class to view all instances. See section 2.2 for a definition of ENUMERATE.
The instance information of all available SELRecordLogCapabilities will be returned.

15.8 **List System Event Log Entries**

Use the following procedure below to list the inventory of all SELLogEntry instances in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetSystemEventLogEntries.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSystemEventLogEntries(): **ENUMERATE** the DCIM_SELLogEntry class to view all instances. See section 2.2 for a definition of **ENUMERATE**.

The instance information of all available SELLogEntry will be returned.

16 **Role-based Authorization Profile (RBAP) Use Cases**

16.1 **Discovery of RBAP Profile Support**

Use the following procedure below to confirm the existence of Role Based Authorization profile support. **NOTE:** Prior to LC2.0.0, this profile resided as a CIM profile, not LC profile.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetRBAPProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) **[LC1.5.0/LC1.5.1]** GetCIMRegisteredProfiles():
   **[LC2.0.0]** GetLCRegisteredProfiles():

      **ENUMERATE** the applicable class to view all registered profiles. See section 2.2 for a definition of **ENUMERATE**.

C) Search for “RegisteredName= Role Based Authorization” and note its instanceID to use in step D)

D) **[LC1.5.0/LC1.5.1]** GetCIMRegisteredProfile():
**[LC2.0.0] GetLCRegisteredProfile():**

GET the applicable instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:RoleBasedAuthorization:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
[DCIM_LCRegisteredProfile]
    AdvertiseTypeDescriptions = WS-Identify
    AdvertiseTypeDescriptions = Interop Namespace
    AdvertiseTypes = 1
    AdvertiseTypes = 1
    InstanceID = DCIM:RoleBasedAuthorization:1.0.0
    OtherRegisteredOrganization = DCIM
    ProfileRequireLicense
    ProfileRequireLicenseStatus
    RegisteredName = Role Based Authorization
    RegisteredOrganization = 1
    RegisteredVersion = 1.0.0
```

### 16.2 Discovery of Users with Assigned LAN Privileges

Enumerate the DCIM_IPMIRBAIdentityMemberOfCollection class to view all IPMI LAN identities that are assigned a role.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetUsersAssignedLANPrivileges.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetUsersAssignedLANPrivileges(): ENUMERATE the DCIM_IPMIRBAIdentityMemberOfCollection class to view all instances. See section 2.2 for a definition of ENUMERATE.

The associations between all LAN Identities and IPMI Roles will be returned.

### 16.3 Discovery of Users with Assigned Serial over LAN Privileges

Enumerate the DCIM_IPMISOLRBAIdentityMemberOfCollection class to view all serial identities that are assigned a role.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetUsersAssignedSerialOverLANPrivileges.win
A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetUsersAssignedSerialOverLANPrivileges(): ENUMERATE the DCIM_IPMISOLRBAIdentityMemberOfCollection class to view all instances. See section 2.2 for a definition of ENUMERATE.

The associations between the serial Identities and IPMI SOL Role will be returned.

16.4 Discovery of users with Assigned CLP Privileges

Enumerate the DCIM_CLPRBAIdentityMemberOfCollection class to view all CLP identities that are assigned a role.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetUsersAssignedCLPPrivileges.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetUsersAssignedCLPPrivileges(): ENUMERATE the DCIM_CLPRBAIdentityMemberOfCollection class to view all instances. See section 2.2 for a definition of ENUMERATE.

The associations between all CLP Identities and CLP Roles will be returned.

17 Service Processor Profile Use Cases

17.1 Discovery of Service Processor Profile Support

Use the following procedure below to confirm the existence of Service Processor profile support.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetServiceProcessorProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Service Processor” and note its instanceID to use in step D)
D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM: SystemInfo:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
AdvertiseTypeDescriptions = WS-Identify
AdvertiseTypeDescriptions = Interop Namespace
AdvertiseTypes = 1
AdvertiseTypes = 1
InstanceID = DCIM:ServiceProcessor:1.0.0
OtherRegisteredOrganization = DCIM
ProfileRequireLicense
ProfileRequireLicenseStatus
RegisteredName = Service Processor
RegisteredOrganization = 1
RegisteredVersion = 1.0.0
```

18 Simple NIC Profile Use Cases

18.1 Discovery of Simple NIC Profile Support

Use the following procedure below to confirm the existence of Simple NIC profile support.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetNICProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Simple NIC” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM: SimpleNIC:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
AdvertiseTypeDescriptions = WS-Identify
AdvertiseTypeDescriptions = Interop Namespace
AdvertiseTypes = 1
AdvertiseTypes = 1
InstanceID = DCIM:SimpleNIC:1.0.0
OtherRegisteredOrganization = DCIM
ProfileRequireLicense = Remote Firmware Configuration
```
18.2 **Inventory of NICs in System**

Use the following procedure below to list the inventory of all NICs in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetNICViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) **GetNICViews():** ENUMERATE the DCIM_NICView class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available NICs will be returned.

18.3 **Get the First NIC’s Information**

Use the following procedure to get a single NIC instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

**Applies to:** LC1.5.0+

**Prerequisites for script:**

- Set desired instanceID by editing script

**Script:** GetNICView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) **GetNICView():** GET the DCIM_NICView instance using an instanceID, such as NIC.Mezzanine.2B-1. See Section 2.3 for a definition of GET.

The instance of DCIM_NICView that contains the information on the first NIC will be returned.

18.4 **List all NIC Attributes**

Use the following procedure below to view all available attributes and possible values of all NICs in the system.

**Applies to:** LC1.5.0+
Prerequisites for script: none

Script: GetAllNICAttributes.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetNICAttributes(): ENUMERATE the DCIM_NICAttribute class to view all instances. See section 2.2 for a definition of ENUMERATE.

18.5 Delete Pending NIC Values

Use the following procedure below to delete pending NIC configurations/values set by the setAttribute(s) method.

Applies to: LC1.5.0+

Prerequisites for script:

- Set desired NIC FQDD

Script: DeletePendingNICConfiguration.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetNICViews(): ENUMERATE the DCIM_NICView class to view all available NIC FQDDs.

See section 2.2 for a definition of ENUMERATE.

C) DeletePendingNICConfiguration(): Deletes the pending configuration for a particular NIC, using a target FQDD such as NIC.Embedded.1-1.

A return message of “No pending data present to delete” indicates that there is no pending NIC configuration to delete for the respective FQDD.

18.6 Discovery of NIC Capabilities

Use the following procedure below to determine the capabilities of NIC cards such as QLogic, Broadcom, and Intel.

Applies to: LC1.5.0+

Prerequisites for script:

- none

Script: NICcapEnable.win
A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetNICCapabilities(): ENUMERATE the DCIM_NICCapabilities class to view all instances of the class. See section 2.2 for a definition of ENUMERATE. If instances are returned, go to end of script as the capabilities are published.

C) GetNICViews(): ENUMERATE the DCIM_NICView class to view all available NIC instances. See section 2.2 for a definition of ENUMERATE. If no instances are returned, go to end of script as no NICs are present.

D) GetSystemViews(): ENUMERATE the DCIM_SystemView class. See section 2.2 for a definition of ENUMERATE. If the “PowerState” field is equal to 8, power on system.

E) RequestPowerStateChange(): Power on the system using PowerState=2.

F) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

* When the system is ready, continue

G) GetNICCapabilities(): ENUMERATE the DCIM_NICCapabilities class to view all instances of the class. See section 2.2 for a definition of ENUMERATE. If instances are returned, go to end of script as the capabilities are published.

H) RequestPowerStateChange(): Power off the system using PowerState=8.

* NOTE: If an operating system has been installed, the system will boot into it. It may be desired to wait until the OS boot is complete before performing a graceful shutdown.

19  Software Update Profile Use Cases

19.1  Discovery of Software Update Profile Support
Use the following procedure below to confirm the existence of software update profile support.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetSoftwareUpdateProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Software Update” and note its instanceID to use in step D)
D) GetLCRegisteredProfile(): GET the `DCIM_LCRegisteredProfile` instance using the `InstanceID` from C). See Section 2.3 for a definition of GET.

Results for the `InstanceID` of DCIM: SoftwareUpdate:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  AdvertiseTypes = 1
  InstanceID = DCIM:SoftwareUpdate:1.0.0
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense = Remote Firmware Update
  ProfileRequireLicenseStatus = LICENSED
  RegisteredName = Software Update
  RegisteredOrganization = 1
  RegisteredVersion = 1.0.0
```

20 Job Control Profile Use Cases

20.1 Discovery of Job Control Profile Support

Use the following procedure below to confirm the existence of Job Control profile support.

**Applies to:** LC1.4.0+

**Prerequisites for script:** none

**Script:** GetJobControlProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the `DCIM_LCRegisteredProfile` class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Job Control” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the `DCIM_LCRegisteredProfile` instance using the `InstanceID` from C). See Section 2.3 for a definition of GET.

Results for the `InstanceID` of DCIM:JobControl:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  AdvertiseTypes = 1
  InstanceID = DCIM:JobControl:1.0.0
  OtherRegisteredOrganization = DCIM
```
20.2 List all Jobs in Job Store

Getting all the jobs in the job store is a matter of confirming the system is in a ready state and then enumerating the `DCIM_LifecycleJob` class to view all available instances of the class.

**Applies to:** LC1.4.0+

**Prerequisites for script:** none

**Script:** ListAllJobs.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) **ENUMERATE** the `DCIM_LifecycleJob` class to view all jobs. See section 2.2 for a definition of `ENUMERATE`.

20.3 Get a Job’s Information

The URI for getting particular instance information is deterministic (i.e the `InstanceId` will be unique for each instance). Job IDs are usually obtained directly from a concrete job, reboot job, or create target config job operations.

**Applies to:** LC1.3.0

**Prerequisites for script:**

- Set desired job variable by editing script

**Script:** Get1Job.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) Use the instanceID from a concrete job, reboot job, or create target config job operation. Note: Reboot job IDs start with RID_ instead of JID_. See section 2.3 for a definition of GET.

The instance of `DCIM_LifecycleJob` that contains the information on the job will be returned.

20.4 Delete all Jobs from Job Store (job queue) using “JID_CLEARALL”

This workflow deletes all jobs from the job queue using by passing JID_CLEARALL as the jobID.
[LC1.5.x and prior] Running this command also restarts remote services.

**Applies to:** LC1.3.0

**Prerequisites for script:** none

**Script:** DeleteAllJobs.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.

B) DeleteJobQueue(): Delete all jobs from the job store by setting JobID="JID_CLEARALL"

C) [LC1.5.x and prior] The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.

D) [LC1.5.x and prior] The remote service will get reset upon invoking the delete job queue command. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile to determine when the remote service is ready.

The GetRSStatus() method must first poll for ‘reloading’ then poll for ‘ready’, while the GetRemoteServicesAPISStatus() can just poll for ‘ready.’

### 20.5 Delete one Job from job store

This workflow deletes one job from the job store (job queue). The user must specify which job ID or reboot ID to be deleted.

**Applies to:** LC1.3.0

**Prerequisites for script:** Specify job ID to be deleted by editing script

**Script:** DeleteOneJob.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISStatus() method may be used depending on the version of the LC Management registered profile.

B) DeleteJobQueue(): Delete specified job from the job store by setting JobID equal to the job ID (JID_01234) or reboot ID (RID_01234).

C) GetLifecycleJobs(): List all remaining jobs in job store

### 21 Memory Profile Use Cases

#### 21.1 Discovery of Memory Profile Support

Use the following procedure below to confirm the existence of memory profile support.

**Applies to:** LC1.5.0+
Prerequisites for script: none

Script: GetMemoryProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the **DCIM_LCRegisteredProfile** class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Memory” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the **DCIM_LCRegisteredProfile** instance using the **InstanceID** from C). See Section 2.3 for a definition of GET.

Results for the **InstanceID** of DCIM:Memory:1.0.0 shown below. If no instance is returned, the profile is not supported.

**NOTE:** 12G example output shown below

```
DCIM_LCRegisteredProfile
 AdvertiseTypeDescriptions = WS-Identify
 AdvertiseTypeDescriptions = Interop Namespace
 AdvertiseTypes = 1
 AdvertiseTypes = 1
 InstanceID = DCIM:Memory:1.0.0
 OtherRegisteredOrganization = DCIM
 ProfileRequireLicense = Remote Inventory
 ProfileRequireLicenseStatus = LICENSED
 RegisteredName = Memory
 RegisteredOrganization = 1
 RegisteredVersion = 1.1.0
```

21.2 Inventory of Memory in System

Use the following procedure below to list the inventory of all memory DIMMs in the system.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetMemoryViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetMemoryViews(): ENUMERATE the **DCIM.MemoryView** class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available memory DIMMs will be returned.
21.3 Get the first Memory’s Information

Use the following procedure to get a single memory instance. The URI for getting particular instance information is deterministic (i.e. the InstanceID will be unique for each instance).

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

- Set desired instanceID by editing script

**Script:** GetMemoryView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetMemoryView(): GET the DCIM_MemoryView instance using the InstanceID= DIMM.Socket.A1. See Section 2.3 for a definition of GET.

The instance of DCIM_MemoryView that contains the information on the first memory DIMM will be returned.

22 PCI Device Profile Use Cases

22.1 Discovery of PCI Device Profile Support

Use the following procedure below to confirm the existence of PCI Device profile support.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetPCIDeviceProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=PCI Device” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:PCIDevice:1.0.0 shown below. If no instance is returned, the profile is not supported.
23 Sensors Profile Use Cases

23.1 Discovery of Sensor Profile Support

There is currently not a registered Sensor Profile prior to LC2.0. However, the CIM_Sensor class has been implemented, which returns all the sensor views when enumerated.

Use the following procedure below to confirm the existence of Sensors profile support in LC2.0+.

**Applies to:** LC2+

**Prerequisites for script:** none

**Script:** GetSensorProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   
   GetRemoteServicesAPStatus();

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Sensors” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:Sensors:1.0.0 shown below. If no instance is returned, the profile is not supported.

**NOTE:** 12G example output shown below

```
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  AdvertiseTypes = 1
  InstanceID = DCIM:PCIDevice:1.0.0
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense = Remote Inventory
  ProfileRequireLicenseStatus = LICENSED
  RegisteredName = PCI Device
  RegisteredOrganization = 1
  RegisteredVersion = 1.0.0
```


AdvertiseTypes = 1
AdvertiseTypes = 1
InstanceID = DCIM:Sensors:1.0.0
OtherRegisteredOrganization = DCIM
ProfileRequireLicense = Device Monitoring
ProfileRequireLicenseStatus = LICENSED
RegisteredName = Sensors
RegisteredOrganization = 1
RegisteredVersion = 1.0.0

23.2 Inventory of Sensor in System

Use the following procedure below to list the inventory of all sensors in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetSensorViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSensorViews(): ENUMERATE the CIM_Sensor class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available sensors will be returned.

23.3 Sensor Thresholds

Only sensors of Numeric type have thresholds (i.e. sensors of type DCIM_PSNumericSensor and DCIM_NumericSensor). Sensor instances of this type will indicate the supported thresholds in their SupportedThresholds property. They also indicate the settable ones through their SettableThresholds property. ‘Set’ operations can be performed to change the settable thresholds.

**Applies to:** LC1.5.0+

**Prerequisites for script:** Set thresholds by editing script

**Script:** SetSensorThresholds.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
B) GetSensorViews(): ENUMERATE the CIM_Sensor class to view all instances. Check the instance of either DCIM_PSNumericSensor and DCIM_NumericSensor to see if there are any settable thresholds.

C) SetSensorThreshold(): SET the DCIM_PSNumericSensor instance for temperature. Set the LowerThresholdNonCritical and UpperThresholdNonCritical values.

24 Base Server and Physical Asset Profile Use Cases

24.1 Discovery of Base Server and Physical Asset Profile Support

There is not an LC registered Base Server and Physical Asset Profile prior to LC2.0.

Use the following procedure below to confirm the existence of the Base Server and Physical Asset profile support in LC2.0+.

Applies to: LC2+

Prerequisites for script: none

Script: GetBaseServerAndPhysicalAssetProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   GetRemoteServicesAPIStatus();

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName= Base Server and Physical Asset” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:BaseServerandPhysicalAsset:1.0.0 shown below. If no instance is returned, the profile is not supported.

DCIM_LCRegisteredProfile
   AdvertiseTypeDescriptions = WS-Identify
   AdvertiseTypeDescriptions = Interop Namespace
   AdvertiseTypes = 1
   AdvertiseTypes = 1
   InstanceID = DCIM:BaseServerandPhysicalAsset:1.0.0
   OtherRegisteredOrganization = DCIM
   ProfileRequireLicense
   ProfileRequireLicenseStatus
   RegisteredName = Base Server and Physical Asset
   RegisteredOrganization = 1
   RegisteredVersion = 1.0.0
24.2 Discovery of Base Server and Physical Asset Profile Support [LC1.5.1]

There is not an LC registered Base Server and Physical Asset Profile prior to LC2.0. There are however, separate Base Server and Physical Asset profiles.

Use the following procedure below to confirm the existence of the Base Server profile and Physical Asset profile support prior to LC2.0.

**Applies to:** LC1.5.0, LC1.5.1

**Prerequisites for script:** none

**Script:** GetBaseServerAndPhysicalAssetProfilesLC151.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetCIMRegisteredProfiles(): ENUMERATE the DCIM_RegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName= Base Server” and note its instanceID to use in step E)

D) Search for “RegisteredName= Physical Asset” and note its instanceID to use in step F)

E) GetCIMRegisteredProfile(): GET the DCIM_RegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

```
CIM_RegisteredProfile
   AdvertiseTypeDescriptions = WS-Identify
   AdvertiseTypes = 1
   Caption
   Description
   ElementName
   InstanceID = DCIM:CSRegisteredProfile:1
   OtherRegisteredOrganization
   RegisteredName = Base Server
   RegisteredOrganization = 2
   RegisteredVersion = 1.0.0
```

F) GetCIMRegisteredProfile(): GET the DCIM_RegisteredProfile instance using the InstanceID from D). See Section 2.3 for a definition of GET.

```
CIM_RegisteredProfile
   AdvertiseTypeDescriptions = WS-Identify
   AdvertiseTypes = 1
   Caption
   Description
   ElementName
```
24.3 List all CIM Profiles

Use the following procedure below to list all CIM profiles supported on a system.

Applies to: LC1.3.0+

Prerequisites for script: none

Script: GetCIMRegisteredProfiles.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetCIMRegisteredProfiles(): ENUMERATE the DCIM_RegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

25 Video Profile Use Cases

25.1 Discovery of Video Profile Support

Use the following procedure below to confirm the existence of Video profile support.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetVideoProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Video” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.
Results for the InstanceID of DCIM:Video:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  AdvertiseTypes = 1
  InstanceID = DCIM:Video:1.0.0
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense = Remote Inventory
  ProfileRequireLicenseStatus = LICENSED
  RegisteredName = Video
  RegisteredOrganization = 1
  RegisteredVersion = 1.0.0
```

25.2 Inventory of Video in System

Use the following procedure below to list the inventory of all video in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetVideoViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSSstatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetVideoViews(): ENUMERATE the DCIM_VideoView class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available video components will be returned.

25.3 Get the first Video Instance’s Information

Use the following procedure to get a single video instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

**Applies to:** LC1.5.0+

**Prerequisites for script:**

- Set desired instanceID by editing script
Script: GetVideoView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetVideoView(): GET the DCIM_VideoView instance using the InstanceID=Video.Embedded.1-1. See Section 2.3 for a definition of GET.

The instance of DCIM_VideoView that contains the information will be returned.

26 License Management Profile Use Cases

26.1 Discovery of License Management Profile Support

Use the following procedure below to confirm the existence of License Management profile support.

Applies to: LC2+

Prerequisites for script: none

Script: GetLicenseManagementProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus():

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName= License Management” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:LicenseManagement:1.0.0 shown below. If no instance is returned, the profile is not supported.

DCIM_LCRegisteredProfile
AdvertiseTypeDescriptions = WS-Identify
AdvertiseTypeDescriptions = Interop Namespace
AdvertiseTypes = 1
AdvertiseTypes = 1
InstanceID = DCIM:LicenseManagement:1.0.0
OtherRegisteredOrganization = DCIM
ProfileRequireLicense = Licensing Management
27 Power Supply Profile Use Cases

27.1 Discovery of Power Supply Profile Support

Use the following procedure below to confirm the existence of Power Supply profile support.

Applies to: LC1.5.0+

Prerequisites for script: none

Script: GetPowerSupplyProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=Power Supply” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:PowerSupply:2.0.0 shown below. If no instance is returned, the profile is not supported.

DCIM_LCRegisteredProfile
 AdvertiseTypeDescriptions = WS-Identify
 AdvertiseTypeDescriptions = Interop Namespace
 AdvertiseTypes = 1
 AdvertiseTypes = 1
 InstanceID = DCIM:PowerSupply:2.0.0
 OtherRegisteredOrganization = DCIM
 ProfileRequireLicense = Remote Inventory
 ProfileRequireLicenseStatus = LICENSED
 RegisteredName = Power Supply
 RegisteredOrganization = 1
 RegisteredVersion = 2.1.0
27.2 Inventory of Power Supply Units (PSUs) in System

Use the following procedure below to list the inventory of all power supplies in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetPowerSupplyViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetFanViews(): ENUMERATE the DCIM_PowerSupplyView class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available power supplies will be returned.

27.3 Get the first PSU’s Information

Use the following procedure to get a single power supply instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

**Applies to:** LC1.5.0+

**Prerequisites for script:**

- Set desired instanceID by editing script

**Script:** GetPowerSupplyView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISstatus() method may be used depending on the version of the LC Management registered profile.

B) GetPowerSupplyView(): GET the DCIM_PowerSupplyView instance using the InstanceID=PSU.Slot.1. See Section 2.3 for a definition of GET.

The instance of DCIM_PowerSupplyView that contains the information on the first power supply will be returned.

27.4 Get MAC Information

This workflow enumerates DCIM_SystemView and invokes GetMACInfo.
**Applies to:** LC1.5.0

**Prerequisites for script:** none

**Script:** CollectBlades.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSystemViews(): ENUMERATE the **DCIM_SystemView** class. See section 2.2 for a definition of ENUMERATE.

C) GetHostMACInfo(): Invoke GetHostMACInfo [ReturnValue=0]

### 27.5 Get Blade Power

This workflow enumerates DCIM_AssociatedPowerManagementService class.

**Applies to:** LC1.5.0

**Prerequisites for script:** none

**Script:** GetBladePower.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetAssociatedPowerManagementService(): ENUMERATE the **CIM_AssociatedPowerManagementService** class. See section 2.2 for a definition of ENUMERATE.

### 28 System Info Profile Use Cases

#### 28.1 Discovery of System Info Profile Support

Use the following procedure below to confirm the existence of System Info profile support.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetSystemInfoProfile.win
The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName=System Info” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the
InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:SystemInfo:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
AdvertiseTypeDescriptions = WS-Identify
AdvertiseTypeDescriptions = Interop Namespace
AdvertiseTypes = 1
InstanceID = DCIM:SystemInfo:1.0.0
OtherRegisteredOrganization = DCIM
ProfileRequireLicense = Power Budget
ProfileRequireLicense = Remote Firmware Configuration
ProfileRequireLicenseStatus = LICENSED
ProfileRequireLicenseStatus = LICENSED
RegisteredName = System Info
RegisteredOrganization = 1
RegisteredVersion = 1.2.0
```

### 28.2 Inventory of System Info View

Use the following procedure below to list the inventory of all system info in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetSystemInfoViews.win

- GetSystemViews(): ENUMERATE the DCIM_SystemView class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available system info will be returned.
28.3 Get the first System info View’s Information

Use the following procedure to get a single system info instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

**Applies to:** LC1.5.0+

**Prerequisites for script:**

- Set desired instanceID by editing script

**Script:** GetSysteminfoView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSystemView(): GET the **DCIM_SystemView** instance using the **InstanceID**= **System.Embedded.1**. See Section 2.3 for a definition of GET.

The instance of **DCIM_SystemView** that contains the information on the first system info will be returned.

28.4 Inventory of all System Attributes in System

Use the following procedure below to list the inventory of all system attributes in the system using the **DCIM_SystemAttribute** class. The instances from the following classes will be returned: **DCIM_SystemEnumeration, DCIM_SystemInteger, DCIM_SystemString**.

**Applies to:** LC2+

**Prerequisites for script:** none

**Script:** GetSystemAttributes.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus():

B) GetSystemAttributes(): ENUMERATE the **DCIM_SystemAttribute** class to view all instances of the attributes: DCIM_SystemEnumeration, DCIM_SystemInteger, and **DCIM_SystemString**. See section 2.2 for a definition of ENUMERATE .

The instance information of all available system attributes will be returned.
28.5 Get a Single System String Attribute

Use the following procedure below to get a single system attribute in the system.

 Applies to: LC2+

Prerequisites for script: Set desired instanceID by editing script

Script: GetSystemString.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   GetRemoteServicesAPIStatus():

B) GetSystemString(): GET the instance using the desired class’s InstanceID. See Section 2.3 for a definition of GET.

The instance that contains the information on the attribute will be returned.

28.6 Setting and Applying System Attributes

The following procedure sets a system attribute and applies the new configuration.

 Applies to: LC2+

Prerequisites for script: Set desired values by editing script

Script: SetSystemAttribute.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   GetRemoteServicesAPIStatus():

B) GetSystemAttributes(): ENUMERATE the DCIM_SystemAttribute class to view all instances of the attributes from the following classes: DCIM_SystemEnumeration, DCIM_SystemInteger, and DCIM_SystemString. See section 2.2 for a definition of ENUMERATE.

C) Confirm correct FQDD and that the IsReadOnly field is set to false

D) SetSystemAttribute(): Invoke the SetSystemAttribute() method on the DCIM_SystemManagement class. The method will set the PendingValue property of the system attributes to the specified value passed in from the input parameters. [Return value = 0]

   NOTE: Use SetSystemAttribute() to set one attribute and SetSystemAttributes() to set multiple attributes. Both methods take the same input parameters but the types of input parameter are different. SetSystemAttributes() uses string input and SetSystemAttributes() uses arrays of strings for input.
E) CreateSystemConfigJob(): Apply the pending values, pass ScheduledStartTime of TIME_NOW, to invoke the reboot to apply the new attribute values immediately.

   NOTE: This method also allows the user to schedule when to apply the attribute value change. If the schedule is TIME_NOW, the PendingValue will be applied to the CurrentValue of the attribute immediately. There is no reboot required for setting system attributes as indicated in the output parameter RebootRequired.

F) Poll jobstatus for Completed: GET the InstanceID of from E). See section 2.3 for a definition of GET.

G) Repeat C) to confirm successful execution of the method by examining PendingValue property of the attributes that were set by this step.

28.7 Apply Pending System Attribute Values

   See the previous section, Setting and Applying System Attributes, for a use case for this method.

28.8 Delete Pending System Attribute Values

   Use the following procedure below to delete pending configurations/values set by the setAttribute(s) method.

   Applies to: LC1.5.0+

   Prerequisites for script: none

   Script: DeletePendingSystemInfoConfiguration.win

   A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

   B) GetSystemViews(): ENUMERATE the DCIM_SystemAttributes class to view all available FQDDs. See section 2.2 for a definition of ENUMERATE.

   C) DeletePendingSystemConfiguration(): Invoke the DeletePendingSystemConfiguration() method on the DCIM_SystemManagementService class to delete pending configurations on ALL system attributes.

   A return message of “No pending data present to delete” indicates that there is no pending configuration to delete for the respective FQDD.
29 Software Inventory Profile Use Cases

29.1 Instance Diagram

![Instance Diagram](image)

**Figure 4. Software Inventory: Instance Diagram**

29.2 Discovery of Software Inventory Profile Support

Use the following procedure below to confirm the existence of Software Inventory profile support.

**Applies to:** LC1.4.0+

**Prerequisites for script:** none

**Script:** GetSWInventoryProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSSstatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): **ENUMERATE** the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of **ENUMERATE**.

C) Search for “RegisteredName = Software Inventory” and note its instanceID to use in step D)
D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:SoftwareInventory:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
   AdvertiseTypeDescriptions = WS-Identify
   AdvertiseTypeDescriptions = Interop Namespace
   AdvertiseTypes = 1
   AdvertiseTypes = 1
   InstanceID = DCIM:SoftwareInventory:1.0.0
   OtherRegisteredOrganization = DCIM
   ProfileRequireLicense = Remote Inventory
   ProfileRequireLicenseStatus = LICENSED
   RegisteredName = Software Inventory
   RegisteredOrganization = 1
   RegisteredVersion = 1.1.0
```

### 29.3 Inventory of Software in System

Use the following procedure below to list the inventory of all software in the system.

**Applies to:** LC1.4.0+

**Prerequisites for script:** none

**Script:** GetSoftwareInventoryViews.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSoftwareIdentities(): ENUMERATE the DCIM_SoftwareIdentity class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available software will be returned.

### 29.4 Get the Installed BIOS Firmware Inventory

Use the following procedure to get the BIOS installed software instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

**Applies to:** LC1.4.0+
Prerequisites for script: none

Script: GetInstalledBIOSView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSoftwareIdentities(): ENUMERATE the DCIM_SoftwareIdentity class to view all instances. See section 2.2 for a definition of ENUMERATE.

Choose the Software Identity instance with ElementName=“BIOS” and Status=“Installed” to use in C)

C) GetSoftwareIdentity(): GET the DCIM_SoftwareIdentity instance from C). See Section 2.3 for a definition of GET.

The instance of the installed BIOS that contains the information will be returned.

29.5 Get the Available iDRAC Firmware Inventory

Use the following procedure to get the available iDRAC software instance. The URI for getting particular instance information is deterministic (i.e the InstanceID will be unique for each instance).

Applies to: LC1.4.0+

Prerequisites for script: none

Script: GetAvailableiDRACView.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetSoftwareIdentities(): ENUMERATE the DCIM_SoftwareIdentity class to view all instances. See section 2.2 for a definition of ENUMERATE.

[Prior to LC2] Choose the Software Identity instance with ElementName=“iDRAC6” and Status=“Available” to use in C)

[LC2] Choose the Software Identity instance with ElementName=“Integrated Dell Remote Access Controller” and Status=“Available” to use in C)

C) GetSoftwareIdentity(): GET the DCIM_SoftwareIdentity instance from C). See Section 2.3 for a definition of GET.

The instance of the available iDRAC that contains the information will be returned.
30  Simple Identity Management Profile Use Cases

30.1  Discovery of Simple Identity Management Profile Support

There is currently not a registered Simple Identity Management Profile prior to LC2. Use the following procedure below to confirm the existence of Simple Identity Management profile support.

**Applies to:** LC2+

**Prerequisites for script:** none

**Script:** GetSimpleIdentityMGMTProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.
   GetRemoteServicesAPIStatus():

B) GetLCRegisteredProfiles(): **ENUMERATE** the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of **ENUMERATE**.

C) Search for “RegisteredName = Simple Identity Management” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): **GET** the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of **GET**.

Results for the InstanceID of DCIM:SimpleIdentityManagement:1.0.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
   AdvertiseTypeDescriptions = WS-Identify
   AdvertiseTypeDescriptions = Interop Namespace
   AdvertiseTypes = 1
   AdvertiseTypes = 1
   InstanceID = DCIM:SimpleIdentityManagement:1.0.0
   OtherRegisteredOrganization = DCIM
   ProfileRequireLicense
   ProfileRequireLicenseStatus
   RegisteredName = Simple Identity Management
   RegisteredOrganization = 1
   RegisteredVersion = 1.0.0
```
31 LC Management Profile Use Cases

31.1 Discovery of LC Management Profile Support

Use the following procedure below to confirm the existence of LC Management profile support.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetLCManagementProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): **ENUMERATE** the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of **ENUMERATE**.

C) Search for “RegisteredName = LC Management” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): **GET** the DCIM_LCRegisteredProfile instance using the **InstanceID** from C). See Section 2.3 for a definition of **GET**.

Results for the **InstanceID** of DCIM:LCManagement:1.1.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  AdvertiseTypes = 1
  InstanceID = DCIM:LCManagement:1.1.0
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense = Auto Discovery
  ProfileRequireLicense = Part Replacement
  ProfileRequireLicense = Remote Firmware Configuration
  ProfileRequireLicense = Remote Inventory Export
  ProfileRequireLicense = Server Profile Export and Import
  ProfileRequireLicenseStatus = LICENSED
  ProfileRequireLicenseStatus = LICENSED
  ProfileRequireLicenseStatus = LICENSED
  ProfileRequireLicenseStatus = LICENSED
  ProfileRequireLicenseStatus = LICENSED
  RegisteredName = LC Management
  RegisteredOrganization = 1
  RegisteredVersion = 1.4.0
```
31.2 Inventory of LC Management Attributes in system

Use the following procedure below to view all available LC attributes and possible values in the system.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** GetAllLCAttributes.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCEnumerations(): ENUMERATE the DCIM_LCEnumeration class to view all instances. See section 2.2 for a definition of ENUMERATE .

C) GetLCStrings(): ENUMERATE the DCIM_LCString class to view all instances. See section 2.2 for a definition of ENUMERATE.

The instance information of all available LC Management attributes will be returned

31.3 Check and enable (or disable) Collect System Inventory on Restart (CSIOR)

This workflow first checks whether CSIOR is enabled (or disabled), if it is not, then it is set to Enabled (or Disabled). While setting the attribute, it checks both the pending and final values.

**Applies to:** LC1.4.0+

**Prerequisites for script:** none

**Script:** EnableCSIOR.win (or DisableCSIOR.win)

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCEnumerations(): ENUMERATE the DCIM_LCEnumeration class and find the current value of CSIOR by searching for “AttributeName=Collect System Inventory on Restart”. See section 2.2 for a definition of ENUMERATE .

C) SetLCAttribute(): Set the CSIOR attribute if it is Disabled (or Enabled)

D) GetLCEnumerations(): ENUMERATE the DCIM_LCEnumeration class to ensure the pending value of CSIOR is Enabled (or Enabled). See section 2.2 for a definition of ENUMERATE.

E) CreateLCConfigJob(): Apply step C) [ReturnValue=4096]
F) Poll jobstatus for Completed: GET the *InstanceId* of from E). See section 2.3 for a definition of GET.
   NOTE: LC1.x polls for ‘COMPLETED’, while LC2 polls for ‘Completed’

G) GetLCEnumerations(): ENUMERATE the DCIM_LCEnumeration class to ensure the new value of CSIOR is Enabled (or Disabled). See section 2.2 for a definition of ENUMERATE.

31.4 Check Version of Lifecycle Controller (LC)

This workflow enumerates the DCIM_SystemView class and searches for the LifecycleControllerVersion attribute to determine the LC version on the system.

**Applies to:** LC1.5.1+

**Prerequisites for script:** none

**Script:** CheckLCVersion.win

A) GetSystemViews(): ENUMERATE the DCIM_SystemView class. See section 2.2 for a definition of ENUMERATE.

B) Search for the LifecycleControllerVersion attribute, the corresponding value of this attribute is the version of LC.

31.5 Get “Part Firmware Update” Attribute

Use the following procedure to get the Part Firmware Update attribute. The URI for getting particular instance information is deterministic (i.e the *InstanceId* will be unique for each instance)

**Applies to:** LC1.4.0+

**Prerequisites for script:** none

**Script:** GetPartFWUpdateAttribute.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCEnumerations(): ENUMERATE the DCIM_LCEnumeration class to new view all available instances. See section 2.2 for a definition of ENUMERATE.

C) Search for “AttributeName = Part Firmware Update” and note its instanceID to use in step D)

D) GetLCEnumeration(): GET the DCIM_LCEnumeration instance using the *InstanceId* from C). See Section 2.3 for a definition of GET.
The instance containing the attribute will be returned.

### 31.6 Check vFlash License Enablement

Use the following procedure to get the Part Firmware Update attribute. The URI for getting particular instance information is deterministic (i.e the `InstanceId` will be unique for each instance)

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** CheckVFlashLicense.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The `GetRSSstatus()` method or the `GetRemoteServicesAPIStatus()` method may be used depending on the version of the LC Management registered profile.

B) `GetLCEnumerations()`: `ENUMERATE` the `DCIM_LCEnumeration` class to view all available instances. See section 2.2 for a definition of `ENUMERATE`.

C) Search for “AttributeName =Licensed” and note its instanceID to use in step D)

D) `GetLCRegisteredProfile()`: `GET` the `DCIM_LCRegisteredProfile` instance using the `InstanceId` from C). See Section 2.3 for a definition of `GET`.

Check the `CurrentValue` parameter to determine if the system is licensed.

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

### 31.7 Set Configuration to “Auto Discovery Factory Defaults”

Use the following procedure to set the Auto Discovery configuration to factory defaults.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none
**Script:** SetAutoDiscoveryFactoryDefaults.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCEnumerations(): ENUMERATE the DCIM_LCEnumeration class to view all available instances. See section 2.2 for a definition of ENUMERATE.

C) Search for “AttributeName = Discovery Factory Defaults” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

The instance containing the attribute will be returned. If the CurrentValue parameter is “on”, Auto Discovery Factory Defaults is already on, proceed to end.

E) ReInitiateDHS(): Invoke method to set Auto Discovery values

F) Repeat step D) to confirm CurrentValue is now ‘on”

### 31.8 Clear Provisioning Server

Use the following procedure to clear the provisioning server name.

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** ClearProvisioningServer.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) ClearProvisioningServer(): Invokes the ClearProvisioningServer method

### 31.9 Replace Auto Discovery Public Key

Refer to Web Service Interface Guide section 12.6
31.10 Replace auto Discovery Client Certificate, Private key and Password

Refer to Web Service Interface Guide section 12.7.

31.11 Delete auto Discovery Public Key

Refer to Web Service Interface Guide section 12.11.

31.12 Delete auto Discovery Client Certificate, Private Key and Password

Refer to Web Service Interface Guide section 12.8.

31.13 Replace iDRAC Web Server Client Certificate and Private Key

Refer to Web Service Interface Guide section 12.10

31.14 Replace iDRAC Web Server Public Certificate

Refer to Web Service Interface Guide section 12.9

31.15 Insert Comment into Lifecycle Log

Use the following procedure to insert a comment into the LC log.

**Applies to:** LC1.5.0+

**Prerequisites for script:** Insert desired comment by editing script

**Script:** InsertCommentInLCLog.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPISstatus() method may be used depending on the version of the LC Management registered profile.

B) InsertCommentInLCLog(): Invokes the InsertCommentInLCLog method
31.16 **Export and View the Content of the Lifecycle Log**

See section 3.16 of this document.

31.17 **Export and View the Current Hardware Inventory**

This workflow exports the hardware inventory to either an NFS or CIFS share. Approximate time for completion on an 11G system is 5 minutes depending on the system configuration.

**Applies to:** LC1.5.0+

**Prerequisites for script:**

- Set script variables by editing script

**Script:** ExportHWInventory.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) ExportHWInventory(): Invokes the export operation [ReturnValue=4096]

   **NOTE:** The user must set applicable IP address, username, password, filename, and workgroup of the network share.

C) Poll jobstatus for Completed: GET the InstanceID of from D). See section 2.3 for a definition of GET.

31.18 **Export and View the Hardware Inventory as Shipped from the Factory**

This workflow exports the factory configuration inventory to either an NFS or CIFS share. Approximate time for completion on an 12G system is 5 minutes depending on the system configuration.

**Applies to:** LC1.5.0+

**Prerequisites for script:**

- Set script variables by editing script

**Script:** ExportFactoryConfiguration.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.
B) ExportFactoryConfiguration(): Invokes the export operation [ReturnValue=4096]

**NOTE:** The user must set applicable IP address, username, password, filename, and workgroup of the network share.

C) Poll jobstatus for Completed: GET the InstanceID of from D). See section 2.3 for a definition of GET.

### 32 OS Deployment Profile Use Cases

#### 32.1 Discovery of OS Deployment Profile Support

Use the following procedure below to confirm the existence of OS Deployment profile support.

**Applies to:** LC1.3.0+

**Prerequisites for script:** none

**Script:** GetOSDProfile.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSStatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

B) GetLCRegisteredProfiles(): ENUMERATE the DCIM_LCRegisteredProfile class to view all registered profiles. See section 2.2 for a definition of ENUMERATE.

C) Search for “RegisteredName= OS Deployment” and note its instanceID to use in step D)

D) GetLCRegisteredProfile(): GET the DCIM_LCRegisteredProfile instance using the InstanceID from C). See Section 2.3 for a definition of GET.

Results for the InstanceID of DCIM:OSDeployment:1.1.0 shown below. If no instance is returned, the profile is not supported.

```
DCIM_LCRegisteredProfile
  AdvertiseTypeDescriptions = WS-Identify
  AdvertiseTypeDescriptions = Interop Namespace
  AdvertiseTypes = 1
  AdvertiseTypes = 1
  InstanceID = DCIM:OSDeployment:1.1.0
  OtherRegisteredOrganization = DCIM
  ProfileRequireLicense = Remote Operating System Deployment
  ProfileRequireLicenseStatus = LICENSED
  RegisteredName = OS Deployment
  RegisteredOrganization = 1
  RegisteredVersion = 1.4.0
```
32.2 **Unpack and Attach Drivers**

See section 4.3 Boot to networkISO and 4.4 Boot to ISO from vFlash for a comprehensive example.

32.3 **Connect and Attach Network ISO Image**

See section 4.3 Boot to networkISO and 4.4 Boot to ISO from vFlash for a comprehensive example.

32.4 **Disconnect and Detach Network ISO Image**

See section 4.3 Boot to networkISO and 4.4 Boot to ISO from vFlash for a comprehensive example.

32.5 **Get ISO Image Connection Status**

See section 4.3 Boot to networkISO and 4.4 Boot to ISO from vFlash for a comprehensive example. GetNetworkISOImageConnectionInfo() can be invoked from within either of the aforementioned scripts to obtain the status.

32.6 **One-time ISO boot Skip**

Following sequence of CIM Operations shall be used to skip ISO boot once. The SkipISOImageBoot() method allows the BIOS to skip booting to the ISO once and boot normally (boot to the first device in boot list).

**Applies to:** LC1.5.0+

**Prerequisites for script:** none

**Script:** SkipISOImageBoot.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands. The GetRSSstatus() method or the GetRemoteServicesAPIStatus() method may be used depending on the version of the LC Management registered profile.

   **NOTE:** GetRemoteServicesAPIStatus() will return “not ready” if drivers or an ISO is already attached.

B) SkipISOImageBoot(): Invoke the method, successful return value is 0
32.7 Remote File Share (RFS) Use Cases

32.7.1 Connect and Attach Network ISO Image as a USB CD-ROM device via RFS USB end point.

Following sequence of CIM Operations shall be used to connect and attach network ISO Image from the specified network share to the host server as a USB CD-ROM device via RFS USB end point.

**Applies to:** LC2+

**Prerequisites for script:** set network parameters by editing script

**Script:** ConnectRFSISOImage.win

**NOTE:** Check of Lifecycle Controller (LC) remote service state, via GetRemoteServicesAPIStatus(), is not required since ISO is connected using RFS USB endpoint and LC status is not relevant.

A) Confirm RFS (remote file share) is NOT in Detach mode. iDRAC attribute AttachMode value needs to be changed to Attached

B) ConnectRFSISOImage(): Invoke the method, successful return value is 0

C) GetRFSISOImageConnectionInfo(): Invoke the method to view connection information and confirm C) was successful

32.7.2 Disconnect and detach ISO Image exposed via RFS USB end point

Following sequence of CIM Operations shall be used to disconnect and detach the ISO Image that is exposed via RFS USB end point to host server.

**Applies to:** LC2+

**Prerequisites for script:** none

**Script:** DisconnectRFSISOImage.win

**NOTE:** Check of Lifecycle Controller (LC) remote service state, via GetRemoteServicesAPIStatus(), is not required since ISO is connected using RFS USB endpoint and LC status is not relevant.

A) DisconnectRFSISOImage(): Invoke the method, successful return value is 0

B) GetRFSISOImageConnectionInfo(): Invoke the method to confirm the B) was successful

32.7.3 Get RF ISO Image connection Status

The GetRFSISOImageConnectionInfo() method is used to provide the status of the ISO Image connection that has been exposed to the host system.
Applies to: LC2+

Prerequisites for script: none

Script: GetRFSISLOImageConnectionInfo.win

NOTE: Check of Lifecycle Controller (LC) remote service state, via GetRemoteServicesAPIStatus(), is not required since ISO is connected using RFS USB endpoint and LC status is not relevant.

A) GetRFSISLOImageConnectionInfo(): Invoke the method, successful return value is 0

32.8 Boot to Hard Drive (HD)

The BootToHD() method is used to boot the system to the hard drive, even if the hard drive is not first in the boot order.

Applies to: LC1.5.2+

Prerequisites for script: none

Script: BootToHD.win

A) The Lifecycle Controller remote service must be in a “ready” state before executing any other WSMAN commands.

GetRemoteServicesAPIStatus():

B) GetRFSISLOImageConnectionInfo(): Invoke the method, successful return value is 0
33 Appendix

33.1 PYTHON scripts README

33.1.1 Purpose

The goal of recite.py is to provide a simple and fast interface for the Dell Lifecycle Controller API. It has an interactive mode that is useful to run one off commands against a server as well as batch mode to allow automating a sequence of operations.

33.1.2 Requirements

Client:

Windows XP or greater with Windows Remote Management (winrm)

Linux with Web Services for Management client (wsmancli)

Python version 2.4 to 2.6

Server:

Dell 11G servers

iDRAC Enterprise version 1.70 (racks and towers), version 3.21 (blades) or greater

Dell Lifecycle Controller version 1.5 or greater

33.1.3 Command line

```
python recite.py [NAME1=VALUE1 NAME2=VALUE2] ["CMD1" "CMD2" ...] [batch1.win
batch2.win]
```

Set variable $NAME1 to VALUE1, $NAME2 to VALUE2 ...

Execute CMD1, CMD2 in order ...

Execute batch scripts in order and exit ...

Enter interactive mode if no scripts specified

```
python recite.py IP=10.0.0.1,10.0.0.2,10.0.0.3
```

Spawn three instances of recite.py in separate windows, each with IP specified
python recite.py IP=username:password@10.0.0.1
Set $IP, $LOGIN and $PASS with a single assignment

python recite.py IP=username:dell123@10.0.0.1,username:dellam@10.0.0.2
Spawn two instances with specified $IP, $LOGIN and $PASS

python recite.py IP=IP.ini
Load IPs from file, one per line, 10.0.0.1 or user:pass@10.0.0.1, # comments a line

python recite.py IP=10.0.0.1 GetRSStatus GetLifecycleJobs
Execute GetRSStatus and GetLifecycleJobs on specified IP

### 33.1.4 Commands

Most of the API methods exposed by the Dell Lifecycle Controller WSMAN interface are supported.

The script also provides a list of common internal commands to allow for minimal programmatic functionality. These include:-

<table>
<thead>
<tr>
<th>Command</th>
<th>Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td></td>
</tr>
<tr>
<td>Clear</td>
<td></td>
</tr>
<tr>
<td>Context</td>
<td>/</td>
</tr>
<tr>
<td>Count</td>
<td>+</td>
</tr>
<tr>
<td>Find</td>
<td>/</td>
</tr>
<tr>
<td>Gosub</td>
<td>&gt;&gt;</td>
</tr>
<tr>
<td>Goto</td>
<td>&gt;</td>
</tr>
<tr>
<td>If</td>
<td>?</td>
</tr>
<tr>
<td>Log</td>
<td></td>
</tr>
</tbody>
</table>
Use the help command in interactive mode to see further details on all available commands and methods and the required syntax.

Apart from the syntax described in help, commands can also be concatenated on methods. This allows for a cleaner syntax.

E.g.

CreateRAIDConfigJob Target=$ctlr RebootJobType=3 {ReturnValue=4096 /$jid=InstanceID
Perform Until loop
On success, perform Find operation

GetRSSStatus {Status=Reloading {Status=Ready
Perform Until looking for Status=Reloading
Perform Until looking for Status=Ready

GetLifecycleJobs +$njob ?$njob=1 >End
Count number of jobs
If only one job, Goto End

Script execution is terminated if:
- A command returns an error. E.g. Find, Context, etc.
- A method returns no data. E.g. GetPhysicalDiskViews when no disks are present.
- A command has a syntax error
In order to ignore such errors and resume execution, prepend command with a `-.

For example:

- Find InstanceID $id
- GetPhysicalDiskViews

### 33.1.5 Settable variables

The following variables are loaded from environment variables if available. If not, they are default initialized as specified.

$IP

IP of the iDRAC against which WS-MAN commands are to be executed. Default: 

Set $IP 10.0.0.1
Set $IP username:password@10.0.0.1

$LOGIN

iDRAC username with WS-MAN privileges. Default: username

Set $LOGIN username

$PASS

iDRAC password. Default: password

Set $PASS dell123

$TIMER

If True, display time taken by WS-MAN command. Default: False

Set $TIMER True

$USLEEP

Default sleep delay in seconds used by until commands between method invocations.
Set $USLEEP 20

$UTIMEOUT
Default total delay in seconds used by until commands before giving up.
Set $UTIMEOUT 900

$VERBOSE
Control level of output from recite.py
Set $VERBOSE x

where x is:
0: Quiet
1: WS-MAN
2: Full

33.1.6 Internal variables

$_BATCHFILE
Name of current batch file (including path) with \ and / replaced with _.

$_DATE
Current date and time in yyyymmddhhmms format.

$_LOCALIP
IP of the local system where script is running.

$_LINE
Current line number in a batch script.
33.2 System check information

33.2.1 Check System Power State
- System power status is available from CMC (racadm/WS-MAN), iDRAC (racadm) and LC.
  - Enumerate the CIM_ComputerSystem class to get power status from LC as described in section 8.2 of the WSIG for Windows
  - If system is not in a state required:
    o Error to user stating system is not in power state required

33.2.2 Check RS status
- Check that \texttt{Status = Ready} as described in section 20 of the WSIG for Windows
  - Poll every 30 seconds until \texttt{Status} changes as required
  - If target system is not Ready, wait for up to 5 minutes before giving up or intervening
  - On timeout, options are:
    o Error to user stating remote services are not ready
    o Reset the iDRAC as described in section 8.4 of the WSIG for Windows

33.2.3 Check for pending jobs
- Enumerate all jobs as described in section 10.2.3 of the WSIG for Windows and look for any jobs (besides \texttt{JID_CLEARALL}) in an incomplete state - i.e. neither \texttt{Completed} nor \texttt{Reboot Completed} state.
  - If any such jobs found:
    - Invoke \texttt{JID_CLEARALL} as described in section 10.2.2 of the WSIG for Windows
    - Poll \texttt{GetRSStatus} for \texttt{Status = Ready} every 30 seconds as described in section 20 of the WSIG for Windows

33.2.4 Check for pending configuration
- Inventory NIC and BIOS attributes as described in sections 33.3.3 and 33.3.5.
  - If any pending configuration found - i.e. \texttt{PendingValue} for any attribute is not blank:
    o Invoke \texttt{DeletePendingConfiguration} on the FQDD in question as described in sections 15.6, 16.15 and 17.8 of the WSIG for Windows

33.2.5 Check CSIOR state
- Get CSIOR status as described in section 12.3 of the WSIG for Windows
  - If CSIOR = \texttt{Disabled}:
    o Set CSIOR = \texttt{Enabled} as described in section 12.1 of the WSIG for Windows
    o Reboot the host using \texttt{RequestStateChange} as described in section 8.4 of the WSIG for Windows for fresh inventory to be collected

33.3 Inventory information

33.3.1 System inventory
- Enumerate \texttt{DCIM_SystemView} class to identify server type and total memory
33.3.2 Software inventory
- Enumerate DCIM_SoftwareIdentity class for all firmware version levels

33.3.3 BIOS inventory
- Enumerate DCIM_BIOSEnumeration class for BIOS attributes of interest

33.3.4 Boot order inventory
- Enumerate DCIM_BootSourceSetting for current boot order settings

33.3.5 NIC inventory
- Enumerate DCIM_NICView class for total number of NICViews, their types and FQDDs. Note that each NICView represents one port (or partition for a partitioned device). Number of NIC controllers can be calculated by trimming out the second and third octet of the FQDD.
  - NIC.Mezzanine.2B-1-1 and NIC.Mezzanine.2B-1-2 are two partitions on the same controller: NIC.Mezzanine.2B
  - NIC.Slot.1-1 and NIC.Slot.1-2 are two ports on the same controller: NIC.Slot.1
- DCIM_NICEnumeration, DCIM_NICString and DCIM_NICInteger classes for all NIC attributes

33.3.6 RAID inventory
- Enumerate DCIM_ControllerView class for total number of RAIDs

33.4 Poll LC jobs information
There are various tasks that are executed in sequence when running one or more LC jobs. Once the job is scheduled using ScheduledStartTime, note that the machine only reboots if a RebootJobType is specified in the CreateTargetedConfigJob method invocation or a reboot job is created using CreateRebootJob and scheduled using SetupJobQueue.

33.4.1 Timing considerations
- While polling jobs for completion, the following target machine specific variables need to be considered when setting reasonable timeouts for job completion:
  - Amount of RAM in machine
    - Influences time taken for initial memory check
    - Details obtained from enumeration described in section 5.3
  - Number of NIC and RAID devices
    - Influences time taken for:
      - POST - controller initialization
      - UEFI initialization - controller driver load and start for CSIOR and SSM
      - Inventory - for CSIOR and SSM
    - Details obtained from enumerations described in sections 33.3.3 and 33.3.5

33.4.2 Machine reboot
- If the machine is turned off, it is turned on in order to execute the jobs.
- If a machine is turned on, it is powercycled at the ScheduledStartTime depending on the RebootJobType specified.
- Once all jobs are successfully scheduled, the target machine power state can be polled as
described in section 5.2.1
- The total time taken for reboot depends on the RebootJobType specified.
  o For a machine powered down, it shouldn’t take more than 30 seconds to power up.
  o For a running machine, timeout depends on the RebootJobType:
    ▪ 1 = PowerCycle - 30 seconds
    ▪ 2 = Graceful Reboot without forced shutdown - 5 minutes
    ▪ 3 = Graceful reboot with forced shutdown - 5 minutes

33.4.3 POST

33.4.3.1 Memory check
- Get the value of the BIOSEnumeration attribute MemTest, obtained during inventory as described in section 5.3.3.
- If MemTest = Enabled, on power on, the BIOS memory test is executed.
- If memory test is executed, the total timeout for POST should include 10 seconds per GB of RAM to check the total RAM installed on the target system, as obtained from enumeration described in section 33.3.

33.4.3.2 Controller initialization
- Get the total number of NIC and RAID controllers on the target system, obtained from enumerations described in sections 33.3.5 and 33.3.6.
- The total timeout for POST should include 10 seconds to initialize each NIC controller installed on the target system.
- The total timeout for POST should include 1 minute to initialize each RAID controller installed on the target system.

33.4.4 SSM

33.4.4.1 UEFI initialization
- Before executing SSM, the UEFI environment needs to load into memory. This includes various BIOS, NIC, RAID and LC drivers that are loaded and started. This also includes initialization time for various NIC device drivers which populate the HII database with their attribute configuration for consumption by LC.
- The total timeout for SSM should include 2 minutes to load all BIOS and LC components.
- The total timeout for SSM should include 30 seconds to load and initialize each NIC driver installed on the target system.

33.4.4.2 Job execution
- Once SSM starts to execute a job, JobStatus will transition from Scheduled to Running to Completed, Completed with Errors, or Failed.
- The total timeout for SSM should include 5-10 minutes per job executed depending on the type of job.
- Error to user if JobStatus != Completed or timeout occurs.
- If JobStatus = Completed with Errors, attributes that failed to set can be obtained from the LCL as described in section 12.13 of the WSIG for Windows.
- Completion of all jobs from the JobStatus perspective will suggest that all job execution tasks are done. However, various LC tasks such as inventory and refresh will still be in progress. The timeouts for these tasks can be accounted for in the next section.
- JobStatus can be obtained as described in section 10.2.3 of the WSIG for Windows.
33.4.4.3 Inventory
- After all jobs are executed, SSM will re-inventory the system and signal the iDRAC to re-sync its database.
- The total timeout for SSM should include 1 minute to inventory each NIC FQDD on the target system after job completion.
- The total timeout for SSM should include 30 seconds to inventory each RAID controller installed on the target system after job completion.

33.4.4.4 Cleanup
- The total timeout for SSM should include 2 minutes to unload and cleanup the UEFI environment prior to reboot.

33.4.5 RSStatus/JobStatus
- After SSM completes, the system will reboot and proceed with CSIOR, as described in 33.4.7.
- In parallel, the iDRAC will refresh its database based on updated inventory.
- The total timeout for RSStatus/JobStatus should include 5 minutes to obtain the inventory data generated and update the iDRAC database.
- Check Status = Reloading and then Status = Ready as described in section 20 of the WSIG for Windows.

33.4.6 Check refreshed data
- Once RSStatus/JobStatus are Ready/Completed respectively, NIC and BIOS need to be re-inventoried to ensure that all pending data was cleared and the set values were successfully applied.
- NIC and BIOS inventory are described in sections 33.3.3 and 33.3.5.
- Error to user stating failures if any pending values still persist or if set values weren’t applied as required.

33.4.7 CSIOR
- On the server side, the following tasks are redone since the machine will reboot and perform CSIOR.
- The overall timeout for SSM should include the following timeouts:
  o POST - as described in section 33.4.3
  o CSIOR - UEFI Initialization, similar to section 33.4.1
  o CSIOR - Inventory
  o CSIOR - Cleanup

33.5 iSCSI boot information
In addition to the programmatic steps shown in section 4.16, additional detailed information about the script is added below.

1. Check FQDD if LOM/NDC or add-in
   o On 11G
     • Embedded = LOM
     • Integrated = LOM
   o On 12G
     • Embedded = LOM
     • Integrated = NDC
   o Addins can be Slot, Mezzanine, etc.
2. Check LOM/NDC enablement
For LOM
- EmbNic1Nic2 needs to be enabled for 1-1, 2-1
- EmbNic3Nic4 needs to be enabled for 3-1, 4-1
- And so forth

For NDC
- IntegratedNetwork1 needs to be enabled for 1-1, 1-2
- IntegratedNetwork2 needs to be enabled for 2-1, 2-2
- And so forth

Addins
- Cannot be disabled on 11G
- On 12G, individual slots can be disabled, TBD if this needs to be addressed

Create a BIOS job
- EmbNic1Nic2 = Enabled or IntegratedNetwork1 = Enabled
- BootMode = Bios
- For Broadcom NICs
  - Disable all IPL boot sources
    - Since target NIC is disabled, it isn’t in the boot list
    - We are doing a BIOS job anyway, so might as well remove other boot sources
  - Wait for job completion
  - Wait for CSIOR to complete
    - Since boot source changes only occur after reboot, they get detected only after CSIOR and refresh

3. Check if FQDD is present in NICView enumeration
   - If NIC is still not present, it is an invalid NIC FQDD, exit

4. Look for NIC in boot sources
   - For Broadcom NICs
     - Look for IPL and NIC FQDD in InstanceIDs in boot source enumeration
     - If FQDD not in boot sources
       - Create a BIOS job
         - BootMode = Bios if not already done
         - Disable all IPL boot sources if not already done
           - Since target NIC isn’t in boot list yet, we can remove other boot sources
       - Create a NIC job
         - LegacyBootProto = iSCSI
           - Adds Broadcom device into boot list in iSCSI boot mode
         - Set all iSCSI boot configuration attributes
       - Wait for both jobs to complete
       - Wait for CSIOR to complete
         - Since boot source changes only occur after reboot, they get detected only after CSIOR and refresh

5. Check if NIC is primary bootable device in boot order
   - For Broadcom NICs
     - Verify if NIC will boot first
       - Get EnabledState of IPL entry for NIC FQDD in boot sources
         - Needs to be enabled, if not, need to do a BIOS job
       - Get assigned sequence number
         - If not first, need to ensure all preceding sources are disabled
         - If other sources will boot before NIC FQDD, need to do a BIOS job
     - If not enabled or a secondary device in boot order
       - Create a BIOS job
         - Set EnabledState for IPL entry of NIC FQDD to 1
         - Set assigned sequence to 0 so that it boots first
• Create a NIC job
  o Set all iSCSI boot configuration attributes
• Wait for both jobs to complete
• Wait for CSIOR to complete
  o Since boot source changes only occur after reboot, they get detected only after CSIOR and refresh

6. iSCSI configuration settings
  o For Broadcom
    • IscsiViaDHCP = Disabled
    • TcpViaDHCP = Disabled
    • ConnectFirstTgt = Enabled
    • IscsiInitiatorIpAddr = <IP of initiator>
    • IscsiInitiatorSubnet = <Subnet mask for initiator>
    • IscsiInitiatorGateway = <Gateway for initiator>
    • IscsiInitiatorName = <Initiator name>
    • IscsiTgtBoot = Enabled
    • FirstHddTarget = Disabled
    • FirstTgtIpAddress = <IP of target>
    • FirstTgtIscsiName = <Target name>
    • FirstTgtBootLun = <Target LUN>

7. System should boot into iSCSI as needed