Hardware Inventory Viewing Script

This Dell technical white paper provides information about using scripts with the WSMAN interface to view the hardware inventory of Dell PowerEdge servers with iDRAC.

Author(s)
Henry Gbedemah
Santosh Bidaralli
Contents

Executive summary .......................................................... 4
Introduction ........................................................................ 4
Environmental requirements ........................................... 4
Using the Script ................................................................. 6
Displaying hardware inventory ........................................ 7
The difference between the factory and currently installed inventory ................. 7
Cleaning up the environment or changing the target iDRAC ................................. 10
Important error information ............................................... 11
Where to find more information ........................................ 11
Summary ............................................................................. 12
Executive summary

This document is written for system administrators who view the hardware inventory of Dell PowerEdge™ servers or who need to compare the current server configuration with what was installed at the factory.

Introduction

Dell PowerEdge servers equipped with Integrated Dell Remote Access Controller (iDRAC) provide secure, simple, scriptable, and standards-based remote management capability through Web Services for Management (WSMAN). WSMAN is a network-based management protocol that lets you access systems management data objects and methods supported by the target platform. Use the WSMAN API with command-line tools such as Windows winrm and Linux wsmancli. This document uses the Python scripting language, which runs on both Windows and Linux. Some reading of specification documents may be required to understand the terminology and concepts in this document.

Environmental requirements

Before you start, prepare the following:

1. Verify your target system is a Dell PowerEdge server with iDRAC enabled, configured, accessible through the network, and reachable by the WSMAN API.

2. Verify Python version 2.7 is installed on your system. If you need help with this, refer to the python release site.

3. Verify that you have the correct hardware configuration.

4. Download the Python scripts package from Dell Tech Center. The scripts relevant to this document are:

   - hardware.py
   - provider_common.py
   - wsman_common.py
Figure 1 shows the environment. It starts with the administrator (1) running scripts to send WSMAN commands through an SSL connection. The target system (2) is a Dell PowerEdge 11th generation server or later equipped with an iDRAC service processor. Use the network share (3) to copy the factory installed hardware configuration file from the target system (capability (K) below).

The hardware Inventory feature displays the following capabilities:

a. Power supply inventory
b. Fan inventory
c. Platform physical memory
d. Platform CPUs
e. iDRAC remote access card
f. Platform PCI devices
g. Platform video controllers
h. Physical attributes of virtual flash media
Hardware Inventory Viewing Script

i. NIC controller configuration

j. System attributes

k. The difference between the factory and current configuration

Using the Script

Use the `hardware.py` script for displaying the capabilities described above.

The `hardware.py` script requires passing a single argument. The script package contains the README_HWInv file, which provides more information about scripts and script usage.

If you run the `hardware.py` script without passing an argument, or if you use an invalid argument, this results in the Sample Usage output/ below.

Sample Usage output:

```
[hardware.py]
Usage: ./hardware.py AttrService
The attribute service AttrService may be one of the following:
power = Power Supply Inventory
fan = Fan Inventory
memory = Platform physical Memory
cpu = Platform CPUs
idrac = iDRAC Remote Access Card
pci = Platform PCI devices
video = Platform Video controllers
vflash = Physical attributes of virtual flash media
nic = NIC Controller Configuration
system = System Attributes
all = All of the installed services
cleanenv = Cleans the working Directory
```
Displaying hardware inventory

When passing a valid argument, such as the `fan` argument, the script prompts for the IP address of the target iDRAC, the user name, and the user password as shown below:

```
[hardware.py fan]
Hardware Inventory
    Enter Target IP Address: [iDRAC IP]
    Enter User Name: [USER NAME]
    Enter password: [PASSWORD]
```

The script establishes a connection with the iDRAC, performs certificate validation, and then displays the requested hardware inventory instance as shown below:

```
Pinging [iDRAC IP]. Waiting for response. Response received.
Getting SSL Certificate. Waiting for response. Response received

Sending command . . .

Hardware Inventory instances
-----------------------------
(1)
DCIM_FanView
ActiveCooling = true
BaseUnits = 19
CurrentReading = 0
FQDD = Fan.Embedded.1
InstanceID = Fan.Embedded.1
LastSystemInventoryTime = 19700101000000.000000+000
LastUpdateTime = 20000105065331.000000+000
PrimaryStatus = 0
RateUnits = 0
RedundancyStatus = 0
UnitModifier = 0
VariableSpeed = true
```

The difference between the factory and currently installed inventory

When passing the `all` argument, if it is the first request sent to the iDRAC, it requests the iDRAC IP address, credentials, and performs certificate validation. Note that if this is not the first command, the IP Address and username are saved in the remote.cfg file in the working directory, so the IP Address and username may not be requested.

```
[hardware.py all]
Hardware Inventory
```
Hardware Inventory Viewing Script

Enter Target IP Address: [iDRAC IP]
Enter User Name: [USER NAME]
Enter password: [PASSWORD]

The script then establishes a connection with iDRAC:

Pinging [iDRAC IP]. Waiting for response. Response received.
Getting SSL Certificate. Waiting for response. Response received

Sending command . . .

Enter user config file name (Example ExportConfig.xml)
Enter IP address of Export Server: [NFS Server IPAddress]
Enter ShareName (Directory path to mount point) Example /home/nfsshare
Enter Share type (NFS=0, TFTP=1, CIFS=2): [Share type]
Enter user name for share server: [USERNAME]
Enter password for share server: [PASSWORD]
Response received.
Job created to Export Factory Configuration

Polling job status

The factory configuration file is saved in the current working directory by prepending the iDRAC IP address to the name given by the user, for example [iDRAC IP]ExportConfig.xml, where Exportconfig.xml is the name provided by the user. The differences between the current and factory installed inventory is captured in an html diff file named output.html, which is show in Figure 2.
The html diff file shows attributes that were added in the current configuration highlighted in green. The legend shows that changed attributes are highlighted in yellow and deleted attributes highlighted in pink, as shown in Figure 3.
Cleaning up the environment or changing the target iDRAC

When the script is run, some intermediate and output files are saved in the working directory. They include:

- The certificate imported from the iDRAC called cer-[iDRAC IP].cer.
- The factory configuration XML file.
- The output.html file, which is the diff html information between the factory configuration and the current configuration.
- A remote.cfg file is maintained to keep track of the iDRAC IP address and username to help reduce data input when using the script. If the remote.cfg file exists in the current working directory, then the script picks up the IP address and username information so you are not prompted to re-enter this information. If you want to see the hardware inventory from a different system, run the script with the cleanenv argument. This argument cleans up the environment, starts over from scratch, and sends the script to a different target iDRAC.

Note: Before running cleanenv, copy all the artifacts needed from the previous run to a different location, so they are not deleted.

The interaction with the script is shown below:

```
[hardware.py cleanenv]
  ... cleaning directory
  cer-[iDRAC IP].cer
```
Hardware Inventory Viewing Script

response.xml
[iDRAC IP]ExportConfig.xml
hardware.log
output.html
remote.cfg

Important error information

If the script ends with the following error:

Exception:

ExportFactoryConfiguration job is in state “failed”!

MessageID = LC034

Message = Factory Configuration was not found on the system

Then there is no factory installed configuration XML file saved on the iDRAC. When this happens, the difference file is not generated and you need to call your Dell support contact.

The artifacts produced from running the hardware script with the all argument are saved in the current working directory.

Where to find more information

Learn more about hardware inventory as defined by the Dell CIM profile specification:

Learn more about job control as defined by the Dell CIM profile specification:
http://www.delltechcenter.com/page/DCIM+Job+Control+Profile+1.1

WSMAN Interface Guide for Linux:
http://attachments.wetpaintserv.us/BMJk79WsVP3F0jwI50xR_w2088275

WSMAN Interface Guide for Windows:
http://attachments.wetpaintserv.us/utYVFQFaHmnfG_LHEnx1YQ2026735

WSMAN command line open source for Linux (Openwsman):
http://sourceforge.net/projects/openwsman/

WSMAN command line for Windows (Winrm):

All about Lifecycle Controller in iDRAC:
http://support.dell.com/support/edocs/software/smusc/smlc/lc_1_5/index.htm

Visit Dell.com/PowerEdge for more information on Dell’s enterprise-class servers.
Summary

If you want to view the hardware inventory of the iDRAC’s host system or compare the current server configuration with the factory installed configuration, then use the native or opensource command-line tools available in Windows and Linux and the python script working sample described in this paper.

About the authors

Henry Gbedemah is a Senior Platform Software Engineer at Dell. He has been with Dell for about 2.5 years developing innovative standards-based software for enterprise systems management.

Santosh Bidaralli is a Platform Software Engineer II at Dell. He has been with for about 2 years developing innovative standards-based software for enterprise systems management.