Using Microsoft™ Windows® PowerShell CIM Cmdlets with Dell™ iDRAC

This Dell Technical White Paper explains how to remotely manage Dell™ PowerEdge™ servers using Dell iDRAC and Windows CIM Cmdlets

Aditi Satam
Syama Poluri
Windows OS Engineering
Contents
Introduction ................................................................................................................. 4
iDRAC and CIM Integration .......................................................................................... 4
  Optional WSMAN Configuration Settings ................................................................. 4
  Establishing the Session ............................................................................................. 5
iDRAC WSMAN-CIM Commands .................................................................................. 7
Passing Parameters to Methods ................................................................................... 8
Conclusion ................................................................................................................... 10
Introduction

The integrated Dell™ Remote Access Controller with Lifecycle Controller (iDRAC with LC) helps manage, monitor, update and deploy Dell Servers. iDRAC provides robust remote management and configuration options without the use of agents and irrespective of the presence and status of the Operating System. It thus lets you manage the server from anywhere, at any time, enabling easier monitoring, upgrade and troubleshooting on the Dell™ PowerEdge™ servers. The latest version of iDRAC is iDRAC7 which is present in the Dell™ 12th generation servers.

Microsoft Windows Server® 2012 introduces PowerShell version 3.0. The latest version of PowerShell has a module called ‘CIMCmdlets’ that contains tasks directly corresponding to the generic CIM operations. The CIM Cmdlets are imported to PowerShell Integrated Scripting Environment (ISE) and PowerShell by default. These Cmdlets work over WSMAN as well as DCOM providing higher flexibility. This article talks about how to run WSMAN CIM commands over iDRAC on Dell PowerEdge servers.

iDRAC and CIM Integration

iDRAC is used to manage the server remotely. You can run WinRM commands (Microsoft’s implementation of the WSMAN protocol) to send commands to the iDRAC. These commands are used for remote management. For example, you can use the WinRM commands to get BIOS information or to set the RAID configuration. For a complete list of server management options using the WinRM commands, refer to the documentation here.

Server administration and management can be highly simplified using the CIM capabilities in Windows® PowerShell and iDRAC7. The PowerShell CIM Cmdlets provide a simple, intuitive scripting language that maps directly to the CIM/WSMAN operations, thus allowing the IT administrator or developer to manage the Dell PowerEdge servers using simple PowerShell scripts.

You can run the PowerShell CIM Cmdlets to manage iDRAC from Windows Server® 2008 R2 or Windows® 7 client systems by installing Windows Management Framework 3.0, available for download here. Thus, the features - Interoperability and backward compatibility, make the usage of PowerShell CIM Cmdlets really flexible and powerful.

Optional WSMAN Configuration Settings

Let’s now move towards understanding the Cmdlets better. Before establishing a session to the iDRAC, you may change some of the WSMAN configuration settings. These can be set from within PowerShell as described in the following section:

Changing the number of instances retrieved:

cd wsman:\localhost

Set-Item .\MaxBatchItems -Value 100
Using Microsoft™ Windows® PowerShell CIM Cmdlets with Dell™ iDRAC

Allowing unencrypted traffic:

cd wsman:\localhost\Client
Set-Item AllowUnencrypted -Value True

List the computers that are trusted:

Set-Item TrustedHosts -Value *

Basic authentication, where username and password are sent in clear text form:

cd wsman:\localhost\Client\Auth
Set-Item Basic -Value True

Establishing the Session

First, you need to set up a session to the iDRAC. This session is not persistent across OS reboots. To set up the session, you need to set the CIM Session options with the Cmdlet New-CimSessionOption.

Note
To get more information about any CIM Cmdlet, run the help cmdlet help -Name [Name of the cmdlet]. For example: Help -Name New-CimSessionOption

Syntax:


EXAMPLE:

$cimop=New-CimSessionOption -SkipCA -SkipCN -SkipRevocation -Encoding UTF8 -UseSsl

Once the session options are set and saved to a variable, you can go ahead and create a session to the iDRAC. This can be accomplished with the New-CIMSession Cmdlet. When you enter the username, you are prompted for the password.
Using Microsoft™ Windows® PowerShell CIM Cmdlets with Dell™ iDRAC

Syntax:


EXAMPLE:

$IPAddress="IDRAC IP ADDRESS"
$Username="USER"

$session=New-CimSession -Authentication Basic -Credential $Username -ComputerName $IPAddress -Port 443 -SessionOption $cimop

echo $session

OUTPUT:

Id            : 1
Name          : CimSession1
InstanceId    : e8615b82-bb22-4ec4-9377-a86c03ba4995
ComputerName  : 10.*.*.*
Protocol      : WSMAN

Note
You can provide the credentials by creating an object of type PSCredential and provide the username and Password.

$IPAddress="IDRAC IP ADDRESS"
$Username="USER"

$Password=ConvertTo-SecureString "PASSWORD" -AsPlainText -Force

$session=New-CimSession -Authentication Basic -Credential $Credentials -ComputerName $IPAddress -Port 443 -SessionOption $cimop

You need to provide the iDRAC IP Address and the credentials for the system and set up a session with Basic authentication. Once the session is established, you are ready to remotely manage your server!
Note: Setting Execution Policy
Please make sure that before you run the scripts, you ensure the execution policy is set up correctly. The default execution policy is Restricted. You can change it to run scripts downloaded from the Internet only that are signed by a trusted publisher using Set-ExecutionPolicy RemoteSigned. For more information please refer here.

iDRAC WSMAN-CIM Commands
Now let us see how you can send different WSMAN commands using the PowerShell CIM Cmdlets. Let us consider the different types of WSMAN commands you run and the type of parameters that are passed. The following sections indicate how the iDRAC WSMAN commands can be used.

Enumerate an instance
The enumerate command lists out the instances of the Resource URI. You can also limit the number of instances returned by using the filter parameter as shown in the examples in the following sections.

The example in this section indicates how the SEL Logs can be retrieved using the Cmdlet Get-CimInstance to enumerate them.

EXAMPLE:

Query Instances
Enumeration of a resource URI can be done by specifying a query. In this case, the query dialect can be specified using the ‘QueryDialect’ parameter.

EXAMPLE:
$query="select * from DCIM_iDRACCardEnumeration WHERE GroupID='Users.1'"
$queryDialect="http://schemas.microsoft.com/wbem/wscim/1/cim-schema/2/root/dcim/DCIM_iDRACCardEnumeration"
Get-CimInstance -Query $query -CimSession $session -Namespace root/dcim -QueryDialect $queryDialect -ResourceUri $resourceUri

Invoke Method
Invoke a method that is specified by the ‘MethodName’. Provide the object by instantiating the class with the required filter. This example illustrates the ‘GetDriverPackInfo’ method that returns the
embedded driver pack version and list of supported OSes for OS deployment that can be installed on the server.

**EXAMPLE:**

```powershell

Invoke-CimMethod -InputObject $inst -CimSession $session -MethodName GetDriverPackInfo | Format-Custom
```

**Passing Parameters to Methods**

**Unique parameters**

To pass unique parameters to the method, the Argument name and value can be passed as a hash table as illustrated in the below example. The RequestedStateChange method takes the argument, ‘RequestedState’ and its value.

**EXAMPLE:**

```powershell

Invoke-CimMethod -InputObject $inst -MethodName RequestStateChange -CimSession $session -Arguments @{RequestedState="3"}
```

**Attribute name and value pair (Named parameters)**

Here is another example, where the SetAttributes method takes the Target, AttributeName and AttributeValue.

**EXAMPLE:**

```powershell
$properties = @{CreationClassName="DCIM_BIOSService" ; SystemCreationClassName="DCIM_ComputerSystem" ; SystemName="DCIM:ComputerSystem" ; Name="DCIM:BIOSService"}

$keyNames = @{$($properties.Keys)

$keyInst = New-CimInstance -ClassName DCIM_BIOSService -Namespace root/dcim -ClientOnly -Key $keyNames -Property $properties
```
Using Microsoft™ Windows® PowerShell CIM Cmdlets with Dell™ iDRAC

```

```

**Array**

In the case of passing multiple attribute name and value pairs, pass them as separate single dimension arrays with each attribute value corresponding to the attribute name in the other array. In this example, ‘BootMode’ is set to ‘Bios’ and ‘BootSeqRetry’ is ‘disabled’.

```
$p=@("BootMode";"BootSeqRetry")
$p1=@("Bios";"Disabled")

$properties=@{CreationClassName="DCIM_BIOSService" ; SystemCreationClassName="DCIM_ComputerSystem" ; SystemName="DCIM:ComputerSystem" ; Name="DCIM:BIOSService"}

$keyNames = @{$properties.Keys}

$keyInst = New-CimInstance -ClassName DCIM_BIOSService -Namespace root/dcim -ClientOnly -Key $keyNames -Property $properties


Invoke-CimMethod -InputObject $inst -MethodName SetAttributes -Arguments @{$Target="BIOS.Setup.1-1";AttributeName=$p;AttributeValue=$p1} -CimSession $session -ResourceUri "http://schemas.dell.com/wbem/wscim/1/cim-schema/2/DCIM_BIOSService"
```

**Reference parameters**

In order to pass a reference as an argument to a method, use `[ref]$instancevariable`. The instance variable contains the object instantiated from the resource URI. Here, invoke-CimMethod is illustrated.

```
```
Conclusion

This document illustrates the ways by which you can use the **PowerShell CIM Cmdlets with iDRAC** to remotely manage the server. The PowerShell CIM Cmdlets let you harness the powerful systems management capability of the WSMAN CIM iDRAC integration using scripts that are both robust and simple. In addition to simplicity, using the PowerShell CIM Cmdlets can reduce network traffic as these Cmdlets require that only a single session be established with the server whereas WinRM requires session establishment and disconnection with every command to the iDRAC. PowerShell provides the system administrator the ability to create custom scripts for remotely managing the server. Since PowerShell is by default present in the Operating system, you don’t need any third party tools to be installed to use CIM Cmdlets.