Understanding the implementation of the PCSV Profile for console usage

This Dell Technical White Paper provides information about iDRAC implementation of the PCSV profile

Dell Engineering
August 2014

Hari Venkatachalam
Chitrak Gupta
Understanding the implementation of the PCSV Profile for console usage

Performance of network reference architectures discussed in this document may vary with differing deployment conditions, network loads, and the like. Third party products may be included in reference architectures for the convenience of the reader. Inclusion of such third-party products does not necessarily constitute Dell’s recommendation of those products. Please consult your Dell representative for additional information.

Trademarks used in this text:
Dell™, the Dell logo, Dell Boomi™, Dell Precision™, OptiPlex™, Latitude™, PowerEdge™, PowerVault™, PowerConnect™, OpenManage™, EqualLogic™, Compellent™, KACE™, FlexAddress™, Force10™ and Vostro™ are trademarks of Dell Inc. Other Dell trademarks may be used in this document. Cisco Nexus®, Cisco MDS®, Cisco NX-OS®, and other Cisco Catalyst® are registered trademarks of Cisco System Inc. EMC VNX®, and EMC Unisphere® are registered trademarks of EMC Corporation. Intel®, Pentium®, Xeon®, Core® and Celeron® are registered trademarks of Intel Corporation in the U.S. and other countries. AMD® is a registered trademark and AMD Opteron™, AMD Phenom™ and AMD Sempron™ are trademarks of Advanced Micro Devices, Inc. Microsoft®, Windows®, Windows Server®, Internet Explorer®, MS-DOS®, Windows Vista® and Active Directory® are either trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Red Hat® and Red Hat® Enterprise Linux® are registered trademarks of Red Hat, Inc. in the United States and/or other countries. Novell® and SUSE® are registered trademarks of Novell Inc. in the United States and other countries. Oracle® is a registered trademark of Oracle Corporation and/or its affiliates. Citrix®, Xen®, XenServer® and XenMotion® are either registered trademarks or trademarks of Citrix Systems, Inc. in the United States and/or other countries. VMware®, Virtual SMP®, vMotion®, vCenter® and vSphere® are registered trademarks or trademarks of VMware, Inc. in the United States or other...
countries. IBM® is a registered trademark of International Business Machines Corporation. Broadcom® and NetXtreme® are registered trademarks of Broadcom Corporation. Qlogic is a registered trademark of QLogic Corporation. Other trademarks and trade names may be used in this document to refer to either the entities claiming the marks and/or names or their products and are the property of their respective owners. Dell disclaims proprietary interest in the marks and names of others.
Contents

Executive Summary .................................................................................................................................................. 5

1 Introduction ........................................................................................................................................................ 5
   1.1 Why PCSV? ............................................................................................................................................... 5

2 Profile ............................................................................................................................................................... 6
   2.1 Profile Explained........................................................................................................................................ 6
   2.2 PCSV Class Diagram ............................................................................................................................... 6
   2.3 Physical Computer System View and Dell View Classes ........................................................................... 7

3 Implementation .................................................................................................................................................. 8
   3.1 Registration .............................................................................................................................................. 8
   3.2 PCSV Class implementation ................................................................................................................... 10
   3.3 Property Mapping .................................................................................................................................... 10
      3.3.1 Property Mapping for Memory ........................................................................................................ 10
      3.3.2 Property Mapping for the Operating System .................................................................................. 11
   3.4 Indexed Arrays ......................................................................................................................................... 11
   3.5 Associations ............................................................................................................................................ 12
   3.6 Methods ................................................................................................................................................... 13
      3.6.1 Methods: RequestStateChange() ..................................................................................................... 14
      3.6.2 Methods: ModifyPersistentBootConfigOrder () ........................................................................... 15

4 PowerShell and PCSV ................................................................................................................................... 16
   4.1 Get-PCSVDevice ..................................................................................................................................... 16
   4.2 Example ..................................................................................................................................................... 16

5 Conclusion ....................................................................................................................................................... 18
Executive Summary

This technical white paper discusses the implementation of the Physical Computer System View (PCSV) profile on the WSMan stack of the iDRAC in the Dell PowerEdge 13th generation servers. The white paper aims to describe the different parts of the PCSV profile and how the users and consoles can use the profile for a consolidated view of their servers.

1 Introduction

The Dell implementation of the PCSV profile provides a consolidated representation and control surface for managing a Dell PowerEdge server. The server is represented by an instance of the DCIM_PhysicalComputerSystemView class and is accessible using the Web Services for Management (WSMan) protocol. The properties of the DCIM_PhysicalComputerSystemView class reflect the current state and configuration of the server.

- PCSV is a one-stop-shop for accessing information about the managed elements.
- PCSV can be directly accessed by PowerShell cmdlets.
- A console can directly enumerate CIM_PhysicalComputerSystem (the parent of DCIM_PhysicalComputerSystemView) without any prior knowledge of a particular class.
- An instance of the CIM_RegisteredProfile class displays information about the implemented features of the physical computer system view.
- Based on implemented features in the instance of the class, supported methods can be invoked.

1.1 Why PCSV?

Dell PowerEdge servers provide 2 standards based APIs for active remote management:

- Intelligent Platform Management Interface (IPMI)
- Systems Management Architecture for Server Hardware (SMASH)

The PCSV profile was formulated by the Distributed Management Task Force (DMTF) to bridge the two standards by adopting the simplicity of the IPMI standard and the extensibility of the WSMan based management. Microsoft as part of their Data Access Layer (DAL) initiative has introduced PCSV PowerShell cmdlets in Microsoft Windows 8.1 and Windows Server 2012 R2. These cmdlets use PCSV as an abstraction to provide a single view whether you are targeting a server that implements IPMI, SMASH, or just PCSV.
2 Profile

2.1 Profile Explained

The Dell PCSV profile is a derivation of DMTF PCSV profile (DSP-1108). The profile contains only one DCIM_PhysicalComputerSystemView class which is a derivation from the CIM_PhysicalComputerSystemView class. The central class (the main class of the profile which represents the managed device) is registered in the DCIM_ProfileRegistrationProfile. The DCIM_ElementView association class relates the central class to the instances of the implemented classes.

2.2 PCSV Class Diagram

The below diagram depicts the class diagram of the PCSV profile.
### 2.3 Physical Computer System View and Dell View Classes

<table>
<thead>
<tr>
<th>Physical Computer System View</th>
<th>Dell View Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>• PCSV is a one-stop-shop for accessing information on the managed elements.</td>
<td>• Standard view classes in Dell defined profiles are based on Dell modeling.</td>
</tr>
<tr>
<td>• PCSV can be directly accessed by PowerShell commandlets.</td>
<td>• Dell View classes are derived from CIM_View class.</td>
</tr>
<tr>
<td>• A console can directly enumerate CIM_PhysicalComputerSystem without any prior knowledge of a particular class.</td>
<td>• The consoles may not require to enumerate all the View classes for the most commonly used device properties</td>
</tr>
<tr>
<td>• Based on the features implemented certain methods can be invoked.</td>
<td>• View classes do not have methods in Dell implementation.</td>
</tr>
</tbody>
</table>

3 Implementation

In this section we will look at the implementation of the PCSV profile in the Dell SMASH architecture.

3.1 Registration

The central class of the PCSV profile DCIM_PhysicalComputerSystemView is registered in the CIM_RegisteredProfile class in the interop namespace. The following diagram shows an example of the registration of PCSV instance. The ImplementedFeatures property in the RegisteredProfile class depicts the implemented classes mapped to PCSV properties.

*Note: The class name prefix (DCIM) has been removed from the below diagrams*
For a Dell Implementation, enumerating the Registered Profile yields a result

**DCIM_RegisteredProfile**

- AdvertiseTypeDescriptions = WS-Identify
- AdvertiseTypeDescriptions = Interop Namespace
- AdvertiseTypes = 1
- AdvertiseTypes = 1
- ImplementedFeatures = Dell:CPUView, Dell:Chassis, DMTF:SoftwareIdentity,
  Dell:NumericSensor, Dell:ComputerSystem, Dell:LCRecordLog
- InstanceID = DCIM:PhysicalComputerSystemViewProfile:1
- OtherRegisteredOrganization = <Nil>
- ProfileRequireLicense = <Nil>
- ProfileRequireLicenseStatus = <Nil>
- RegisteredName = Physical Computer System View
- RegisteredOrganization = 2
- RegisteredVersion = 1.0.0

Implemented features describe the profiles under the purview of this PCSV implementation
3.2 PCSV Class implementation

The PCSV class implementation is provided under the root/dcim namespace. Only one instance of the PCSV class is instantiated. The properties of the PCSV class are indexed arrays of the properties of the associated implemented classes. For example, the PCSV class which implements the ComputerSystem, CPU and RecordLog features, will have the properties of each of the instances of the implemented classes mapped to its own properties. For more information on property mapping and indexed array representation, see section 3.3 and section 3.4. The PCSV instance is associated with the implemented instances through DCIM_ElementView association class.

3.3 Property Mapping

The following tables show a few properties of device classes as they are mapped to the PCSV class properties.

3.3.1 Property Mapping for Memory
<table>
<thead>
<tr>
<th>PhysicalComputerSystemView property name</th>
<th>Memory property name (original)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemoryBlockSize</td>
<td>BlockSize</td>
</tr>
<tr>
<td>MemoryNumberOfBlocks</td>
<td>NumberOfBlocks</td>
</tr>
<tr>
<td>MemoryConsumableBlocks</td>
<td>ConsumableBlocks</td>
</tr>
</tbody>
</table>

### 3.3.2 Property Mapping for the Operating System

<table>
<thead>
<tr>
<th>PhysicalComputerSystemView property name</th>
<th>OperatingSystem property name (original)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSEnabledState</td>
<td>EnabledState</td>
</tr>
<tr>
<td>OSVersion</td>
<td>Version</td>
</tr>
</tbody>
</table>

### 3.4 Indexed Arrays

Most of the device properties in the PCSV class are indexed arrays. The diagram below shows an example of how indexed arrays must be read. The Numericsensor class has three instances, comprising of similar properties. These properties are mapped directly to the PCSV class as an indexed array. For example, in the Numeric Sensor class there is a property called CurrentReading. In each of the instances the CurrentReading is 90, 120, and 0 respectively. These are in different instances. In the PCSV instance, the CurrentReading property is an indexed array. The value is CurrentReading[ ] = 90, 120, 0 which corresponds to the ElementName=Inlet Sensor, CPU Sensor, Base Board Sensor. Indexed arrays provide 1 to 1 mapping for properties.
3.5 Associations

When one or more objects of the DCIM_Processor are instantiated, each instance will be associated with the DCIM_PhysicalComputerSystemView instance through an instance of the DCIM_ElementView, where the Antecedent property is a reference to the corresponding DCIM_Processor instance and the Dependent property is a reference to the DCIM_PhysicalComputerSystemView instance.
3.6 Methods

Current implementation of PCSV in iDRAC supports the following methods:

- CIM_PhysicalComputerSystemView.RequestStateChange
- CIM_PhysicalComputerSystemView.ModifyPersistentBootConfigOrder()
3.6.1 Methods: RequestStateChange()

The RequestStateChange() method changes the state of the physical computer system to the value specified in the RequestedState parameter.

Method Parameters are provided in the below table.

<table>
<thead>
<tr>
<th>Qualifiers</th>
<th>Name</th>
<th>Type</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN, REQ</td>
<td>RequestedState</td>
<td>uint16</td>
<td>Valid state values (mapped to CIM_ComputerSystem.RequestStateChange())</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 (Enabled) – On</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 (Disabled) – Off-Soft</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11 (Reset) – Power-Cycle (Off-Soft)</td>
</tr>
<tr>
<td>IN</td>
<td>TimeoutPeriod</td>
<td>Datetime</td>
<td>Client specified maximum amount of time the transition to a new state is supposed to take:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 or NULL – No time requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;interval&gt; – Maximum time allowed</td>
</tr>
<tr>
<td>OUT</td>
<td>Job</td>
<td>CIM_ConcreteJob REF</td>
<td>Returned if job started</td>
</tr>
</tbody>
</table>

WinRM Command

The example below discusses changing the power of the server to “On” using the RequestStateChange method from the MicrosoftWinRM tool.

Example:

```
```
3.6.2 Methods: ModifyPersistentBootConfigOrder ()

The ModifyPersistentBootConfigOrder () method changes the boot configuration of the physical computer system to the sequence specified in the method parameters.

<table>
<thead>
<tr>
<th>Qualifiers</th>
<th>Name</th>
<th>Type</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN, REQ</td>
<td>StructuredBootString</td>
<td>string array</td>
<td>An array of StructuredBootString values</td>
</tr>
<tr>
<td>OUT</td>
<td>Job</td>
<td>CIM_ConcreteJob REF</td>
<td>Returned if job started</td>
</tr>
</tbody>
</table>

WinRM Command

Following is an example of the RequestStateChange method from using Microsoft WinRM tool. The parameter list is passed as an XML file, BootOrder.xml.

Example:

```
```

BootOrder.xml

```
  <p:StructuredBootString>IPL:BIOS.Setup.1-1#BootSeq#HardDisk.List.1-1#c9203080df84781e2ca3d512883dee6f</p:StructuredBootString>
</p:ModifyPersistentBootConfigOrder_INPUT>
```
4  Powershell and PCSV

Microsoft has defined a suite of PowerShell cmdlets to work with a PCSV device. The following cmdlets are provided by PowerShell to work with a PCSV device over IPMI or WSMAN. The support for PCSV cmdlets has been introduced in Windows Server 2012 R2 and Windows 8.1.

```
PS C:\> Get-Command -Module PcsvDevice

CommandType          Name                                ModuleName
-----------          ----                                ---------
Function            Get-PcsvDevice                      PcsvDevice
Function            Restart-PcsvDevice                 PcsvDevice
Function            Set-PcsvDeviceBootConfiguration    PcsvDevice
Function            Start-PcsvDevice                  PcsvDevice
Function            Stop-PcsvDevice                   PcsvDevice
```

4.1  Get-PCSVDevice

The following describes a usage of the Get-PcsvDevice cmdlet which returns the instance of the DCIM_PhysicalComputerSystemView class. The cmdlet is generic to run on any managed system which support PCSV profile. The following command when run on iDRAC will return as given in the example below

```
```

4.2  Example

```
PS C:\> $cred = Get-Credential root
PS C:\> $ip = "10.10.10.10"
PS C:\> Get-PcsvDevice -TargetComputerName $ip -ManagementProtocol WSMAN - Credential $cred | Format-List
```
CurrentBIOSBuildNumber : 0
CurrentBIOSMajorVersion : 0
CurrentBIOSMinorVersion : 3
CurrentBIOSRevisionNumber : 3
CurrentBIOSVersionString : 0.3.3
CurrentManagementFirmwareBuildNumber : 0
CurrentManagementFirmwareElementName : Integrated Dell Remote Access Controller
CurrentManagementFirmwareMajorVersion : 2
CurrentManagementFirmwareMinorVersion : 0
CurrentManagementFirmwareRevisionNumber : 0
CurrentManagementFirmwareVersionString : 2.00.00.00
Dedicated : 0
EnabledState : 2
FRUInfoSupported : true
HealthState : 25
IdentifyingDescriptions : {CIM:GUID, CIM:Tag, DCIM:ServiceTag}
InstanceID : srv:system
LogCurrentNumberOfRecords : 1961
LogInstanceID : DCIM:LifeCycleLog
LogMaxNumberOfRecords : 0
LogOverWritePolicy : 8
LogState : 2
Manufacturer : Dell Inc.
Model : PowerEdge R730
NumberOfProcessorCores : 14
NumberOfProcessorThreads : 28
NumericSensorBaseUnits : {2, 2, 2, 19...}
NumericSensorCurrentReading : {200, 210, 530, 1800...}
NumericSensorCurrentState : {Normal, Normal, Normal, Normal...}
NumericSensorElementName : {System Board Inlet Temp, System Board Exhaust Temp, CPU1 Temp, System Board Fan1 RPM...}
NumericSensorEnabledState : {2, 2, 2, 2...}
NumericSensorHealthState : {5, 5, 5, 5...}
NumericSensorLowerThresholdCritical : {-70, 0, 30, 360...}
NumericSensorLowerThresholdNonCritical : {30, 0, 80, 600...}
NumericSensorOtherSensorTypeDescription : {NULL, NULL, NULL, NULL...}
NumericSensorPrimaryStatus : {1, 1, 1, 1...}
NumericSensorRateUnits : {0, 0, 0, 0...}
NumericSensorSensorType : {2, 2, 2, 5...}
NumericSensorUnitModifier : {-1, -1, -1, 0...}
NumericSensorUpperThresholdCritical : {470, 750, 970, 268435455...}
NumericSensorUpperThresholdNonCritical : {420, 700, 920, 268435455...}
OperationalStatus : 6
OtherIdentifyingInfo : {ffffffff-ffff-ffff-ffff-ffffffffffff, mainsystemchassis, Unknown}
PartNumber :
ProcessorCurrentClockSpeed : 1200
ProcessorFamily : B3
ProcessorMaxClockSpeed : 4000
RequestedState : 0
SKU :
SerialNumber :
Tag : mainsystemchassis
PSCOMputerName : 10.94.225.160
For more information on PowerShell PCSV cmdlets, see


5 Conclusion

iDRAC on PowerEdge 13th generation server lineup will support PCSV profile as described in this white paper. PCSV bridges the gap between the efficiency of IPMI with the extensibility of a web services based management protocol. Dell has derived a PCSV profile from the DMTF profile. The Dell profile will be published in the Dell Tech Center.

The DMTF profile is available at

http://dmtf.org/sites/default/files/standards/documents/DSP1108_1%20i.pdf