Converged Network Adapters with NIC, iSCSI and FCoE Support

This Dell Technical White Paper introduces modern CNA devices and demonstrates various configuration scenarios using the Dell Lifecycle Controller Remote Services API

Ganesh Viswanathan

June 2012| Rev 1.0

Contents

Executive summary

The new Converged Network Adapters (CNA) supported by Dell Lifecycle Controller provides an array of networking features converged into a single controller, including:

1. Traditional Network Interface Controller (NIC) capabilities
2. Internet Small Computer System Interface over Ethernet (iSCSI) capabilities
3. Fibre Channel over Ethernet (FCoE) capabilities

This technical whitepaper introduces these features and demonstrates configuring various aspects of these devices using the Dell Lifecycle Controller Remote Services API.

Introduction

This white paper introduces the various capabilities of Converged Network Adapters (CNA) devices. This overview includes concepts such as:

1. Personalities
2. Bandwidth partitioning
3. Boot versus Offload
4. I/O Identity

The configuration scenarios section includes detailed examples of configuring these new capabilities using the Dell Lifecycle Controller Remote Services API.

Introduction to Converged Network Adapters

The challenge of connecting servers and storage for maximum performance and maintainability in a data center has a long history. A variety of approaches have been successful, including direct attached storage which has evolved from Small Computer System Interface (SCSI) to Serial Attached SCSI (SAS), which connects most server-grade hard drives today. Storage that lives on the network is another flexible approach that allows better organization and sharing of resources, and includes ubiquitous media such as Ethernet and Fibre Channel.

Ethernet's affordability and versatility ensured its continuing success in most data centers, except for very high bandwidth pipes which remained the domain of Fibre Channel. As Ethernet networking speeds grew from 1MBps all the way to 10GBps and further, they are proving to be increasingly popular in storage networks that demand high bandwidth and performance, while avoiding expensive Fibre Channel networking infrastructure.

Converged network adapters or CNAs are a new class of networking devices that reflect this phenomenon by adding support for various popular protocols to run over the same Ethernet backbone. This includes continued support for traditional IP-based NIC traffic such as PXE booting. Further, support for iSCSI over Ethernet allows hardware offloaded networking with storage targets connected across standard Ethernet cables.

Going further, CNAs also allow multiple servers to be connected to a Fibre Channel storage target by supporting the Fibre Channel over Ethernet (FCoE) protocol. This allows high end and high bandwidth Fibre Channel storage targets to be networked with an array of servers connected with traditional Ethernet infrastructure, along with switches that extract and direct the FCoE payload to the Fibre Channel attached storage targets. This provides data centers with flexibility and affordability to solve the big data problems of the 21st century.

CNAs also provide another interesting feature which is network partitioning. This concept allows the same port and physical network cable, currently capable of 10GBps speeds, to be divided into multiple partitions, each of which can be assigned a portion of the total bandwidth. Each of these partitions then can be assigned a protocol-level personality of NIC, iSCSI or FCoE which then show up in the operating system as individual controllers that can be assigned specific roles. This allows for even more flexibility and control over the bandwidth distribution, quality of service, and the functional capabilities available to the system.

When a remote storage target is attached to the system, either via PXE, iSCSI or FCoE, it can then be utilized as a boot disk by leveraging the boot capabilities of the CNA, or as a data disk with most of the network processing offloaded to the CNA hardware. This frees up the host system to focus on application-level processing rather than being distracted by the infrastructure.

In addition, the unique identity of each port or partition of the CNA can be configured so that remote storage targets can recognize it and attach a corresponding disk. If the CNA or server were to go down due to power or hardware issues, a backup CNA or server can be assigned the same identity so it can take over the job of the failed device. This redundancy helps maintain a high uptime and provides administrators with backups in place while failing devices are fixed or replaced, without impacting operations.

The following sections go into the details of each of these aspects.

Personalities

Personalities on CNA devices can be defined as the different networking protocols supported by the controller. CNA devices currently support the following personalities:-

1. Traditional Network Interface Controller (NIC)
2. Internet Small Computer System Interface over Ethernet (iSCSI)
3. Fibre Channel over Ethernet (FCoE)

These personalities can be assigned to each port for CNAs that do not support partitioning or if partitioning is turned off. For partitionable CNAs, personalities can be assigned to each partition. Each port or partition can be assigned one or more (depending on the controller) personalities that will be active once the operating system boots up and the CNA driver is loaded. In the operating system, each entity can then be assigned a specific task as required by the configuration.

When a NIC personality is assigned to a port or partition, it shows up as a regular network interface controller that can be assigned an IP address and operates as a regular Ethernet device.

The iSCSI personality allows one or more ports or partitions to show up as hardware-based iSCSI initiators, each of which can be configured in the operating system to connect to remote iSCSI targets. This is extremely useful since it offloads all iSCSI, TCP and Ethernet overhead to the CNA hardware processor rather than using the host CPU which can instead focus on other tasks such as virtual machine or database workloads. Once the storage targets are connected, they show up as regular SCSI devices to the operating system.

Similarly, the FCoE personality configured on ports or partitions allows the offloading of FCoE storage target traffic processing to the CNA hardware processor instead of using the host CPU. Again, offloading the processing to the CNA hardware further ensures maximum storage throughput while allowing the host CPU to focus on workloads.

Booting to a PXE, iSCSI or FCoE target is described separately in the Boot versus Offload section below.

Details on how to assign personalities using the Dell Lifecycle Controller Remote Services API are discussed under the configuration scenarios section.

Bandwidth Partitioning

Each modern CNA port provides 10Gbps of bandwidth per port. It makes sense to allow partitioning that bandwidth across various tasks rather than being forced to allocate it to one task per port. This makes even more sense when we consider that each port requires a dedicated physical Ethernet cable, and ensuring 100% usage of the capacity of each cable makes costs go down and maintenance easier.

The NIC Partitioning feature of CNAs allows administrators to organize multiple tasks on each port and intelligently distribute the full capacity of each port’s total bandwidth across the various partitions.

In theory, NIC Partitioning only splits the bandwidth of a port into multiple portions. However, the concept is analogous with hard drive partitioning, where each partition splits the disk space into multiple portions, but each portion then shows up as a “mini” hard drive that can be treated the same as the full drive itself.

Similarly, each partition of a CNA port behaves like a “mini” port and can take up one or more personalities. Each personality of the partition further shows up as an individual entity which can then be configured to address specific needs in the operating system. Each partition performs according to the portion of bandwidth allocated to it and administrators can ensure the quality of service on certain partitions over others by such intelligent allocation of bandwidth.

There are two aspects to configuring bandwidth allocation: minimum bandwidth and maximum bandwidth.

Minimum Bandwidth

Minimum bandwidth is the absolute minimum percentage of bandwidth that should be allocated to a partition. The CNA has to guarantee this minimum bandwidth to the partition. This means that the sum of minimum bandwidths of all partitions on a port should add up to 100%.

Compared to the hard drive partitioning concept, minimum bandwidth is equivalent to the drive size specified for each hard drive partition.

Minimum bandwidth can be configured to add up to less than 100% on some controllers, similar to unallocated space on a hard drive, but it could mean that some of the port bandwidth is unused and hence wasted. Each CNA vendor can choose different bandwidth management schemes to ensure maximum efficiency of the controller while adhering to the specified configuration.

Maximum Bandwidth

Maximum bandwidth is the relative maximum percentage of bandwidth allocated to a partition. When all partitions of a CNA are used to full capacity, each will get allocated only the minimum bandwidth configured. This will ensure that each partition is assured the bandwidth promised to it via the minimum bandwidth setting.

However, if one or more partitions are not active or used to full capacity at a given moment, the balance free bandwidth available will be distributed across the other active partitions until the maximum bandwidth configured for each partition is reached. Essentially, the free bandwidth available on the port will be distributed across all partitions to ensure maximum efficient usage and performance of the port bandwidth.

The maximum bandwidth setting allows each partition to get additional bandwidth when other partitions are idle. In theory, maximum bandwidth should be configured at 100% to make the most of the port capacity. However, an administrator could configure this setting intelligently in order to adjust quality of service of certain partitions over others. For example, traditional NIC traffic could be assigned a lower maximum bandwidth relative to iSCSI or FCoE storage on the same port to ensure the best storage performance.

Details on how to enable NIC Partitioning as well as to configure bandwidth across the partitions using the Dell Lifecycle Controller Remote Services API are discussed under the configuration scenarios section.

Boot versus Offload

The advantages of offloading storage traffic to the CNA hardware processor was discussed in the section discussing personalities. Booting to a storage target, either via iSCSI or FCoE uses the same offload technology to connect and interact with a remote target. However, it is slightly different from a use case perspective and hence elaborated here separately.

Setting up offloaded storage targets is typically done within the operating system. Since the iSCSI and FCoE hardware initiators show up as configurable devices in the operating system, they can be easily configured to connect multiple storage targets, such as connecting a remote storage target as the data drive for an Oracle database.

Where the operating system resides is where the difference shows up. If a local RAID or SATA/SAS drive is setup is preferred, the administrator can install the operating system on the internal drive itself. All storage targets can be connected to the system using the CNA controller’s offload capabilities and can be done after the operating system boots up and the CNA OS driver is loaded.

However, if the operating system itself resides on a remote drive, the CNA controller needs to be configured to connect this boot drive to the system during POST itself and add the drive to the list of available bootable devices on the system. This remote drive will show up just like any other hard drive connected to the machine and the BIOS boot order can be configured to boot to this remote drive after POST is complete.

Some CNA controllers allow attaching a boot drive from each of the protocols below simultaneously. The BIOS boot order can then be tweaked to pick the actual drive to boot to. However, most CNA controllers allow enabling only one boot protocol at a time since the boot option ROM is shared. The options are as follows:

- Boot from a PXE target in NIC mode
- Boot from an iSCSI target in iSCSI offload mode
- Boot from an FCoE storage target in FCoE offload mode

Realistically, a machine only needs to boot from one target at a time. However, for failover purposes, one could configure two or more ports of one or more CNA devices, to connect to the same or different storage targets for redundancy purposes. Refer to the vendor's manual to evaluate the exact capabilities of a device before attempting to configure it remotely.

iSCSI and FCoE boot configurations can be easily set up by using the Dell Lifecycle Controller Remote Services API that provides comprehensive CNA inventory and configuration features. Configuring such a setup via the Dell Lifecycle Controller Remote Services API is discussed in separate whitepapers.

Note that the current generation of CNA devices and Dell Lifecycle Controller do not allow configuring the post-OS iSCSI and FCoE storage target configuration.

I/O Identity

Another interesting feature of CNA devices, related to iSCSI and FCoE boot exposed by the Dell Lifecycle Controller Remote Services API is the I/O Identity feature. This functionality is similar to the FlexAddress feature provided by the Chassis Management Controller for modular systems.

Each CNA controller ships with a hard-coded set of MAC or media access control addresses, essential building blocks of the Ethernet protocol that are unique to the card. Each port and partition, depending on the CNA controller's capabilities, comes with the following identities:

1. NIC MAC address
2. iSCSI MAC address
3. FCoE Initialization Protocol (FIP) MAC address
4. World Wide Name (WWN)
5. World Wide Port Name (WWPN)

The first three MAC identities are used on the Ethernet network, depending on the personality configured on that port or partition, in order to uniquely identify that port or partition on the network. The WWN and WWPN identities are used in the Fibre Channel side, after FCoE traffic is converted into pure FC traffic on Fibre Channel cables.

For each of these attributes, there is a corresponding “virtual” instance that can be set to a different unique or custom value, so as to assign a new known identity to that particular port or partition. This virtual identity persists on the controller unless the controller loses power due to A/C power loss.

The idea behind this feature is to assign a “known” identity to the CNA port or partition and to assign a corresponding storage target for that identity by configuring network infrastructure such as DHCP servers for iSCSI or fabrics and storage arrays for FCoE. As long as the CNA controller is active and functional, it will own that storage target due to the assigned identity. If the system is taken down or fails due to power or hardware issues, another node can be assigned the same identity and it can take over and resume the workload assigned to that identity.

Virtual identities are read-only when accessed directly on the local machine, and can only be set via the Dell Lifecycle Controller Remote Services API. This is because an individual machine does not have the context of the entire network and an external entity such as a management console will typically be assigning identities and workloads, and correspondingly moving workloads if any individual server goes offline.

Note that the FlexAddress feature needs to be turned off on the Chassis Management Controller for modular systems before the I/O Identity feature can be leveraged. Since both features affect the same identities of the CNA, they are not intended to be used in parallel.

More details on how to exercise the I/O Identity feature using the Dell Lifecycle Controller Remote Services API are discussed under the configuration scenarios section.

Configuration Scenarios

Documentation

The following scenarios detail how to interact with CNA devices using the Dell Lifecycle Controller Remote Services API. For details of the entire CNA API such as methods and properties available, the best place to start is the profile documentation. They explain “what” is supported by Dell Lifecycle Controller provided within the context of the CIM architecture.

1. Start at the Dell TechCenter: http://delltechcenter.com/lc
2. Under Reference Specifications, find the link to all Profiles
3. The DCIM Simple NIC Profile documents all the CNA specific capabilities

A complement to the profile documentation is the MOF files that document the class implementation of the Dell-specific CIM classes.

1. Start at the Dell TechCenter
2. Under Reference Specifications, find the link to MOFs
    - The following classes are of interest from a CNA perspective
        - DCIM_NICAttribute
        - DCIM_NICCapabilities
        - DCIM_NICEnumeration

- DCIM_NICInteger
- DCIM_NICService
- DCIM_NICStatistics
- DCIM_NICString
- DCIM_NICView

Figuring out "how" to use the Dell Lifecycle Controller WS-MAN interface is the next step and best documented in the Web Services Interface Guide.

1. Start at the Dell TechCenter
2. Under Web Services Integration Tools, you can find links to the Windows and Linux WSIG
    - Section 15 covers all CNA related commands
    - Sample scripts corresponding to all sections in the documents are also provided, using winrm on Windows and wsmancli on Linux.

Workflows that group various atomic WS-MAN operations to achieve a particular task are the next step and best documented in the Best Practices Guide.

1. Start at the Dell TechCenter
2. Under Web Services Integration Tools, find the link to the Best Practices Guide
    - Sample scripts corresponding to all sections in the document are also provided, using winrm on Windows and wsmancli on Linux

NIC Inventory

To start out, we first need to know what CNA devices are available on our target machine. This can be done by enumerating the `DCIM_NICView` class and identifying the `InstanceID` of each entry.

```
winrm enumerate "cimv2/root/dcim/DCIM_NICView" -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic
```

```
DCIM_NICView
  AutoNegotiation = 3
  BusNumber = 1
  ControllerBIOSVersion
  CurrentMACAddress = 14:FE:B5:1A:40:5E
  DataBusWidth = 0002
  DeviceNumber = 0
  EFIVersion
  FCoEOffloadMode = 3
  FCoEWWNN
  FQDD = NIC.Integrated.1-1-1
  FamilyVersion = 01.09.22
  FunctionNumber = 0
  InstanceID = NIC.Integrated.1-1-1
  LastSystemInventoryTime = 20120523134752.000000+000
  LastUpdateTime = 20120523134738.000000+000
  LinkDuplex = 0
  LinkSpeed = 5
  MaxBandwidth = 100
  MediaType = 2
  MinBandwidth = 0
  NicMode = 2
  PCIDeviceID = 8020
  PCISubDeviceID = 1f64
  PCISubVendorID = 1028
  PCIVendorID = 1077
  PermanentFCOEMACAddress
  PermanentMACAddress = 00:0E:1E:0B:9A:30
  PermanentiSCSIMACAddress
  ProductName = QLogic CNA 10 Gigabit Ethernet QMD8262 -
14:FE:B5:1A:40:5E
  ReceiveFlowControl = 2
  SlotLength = 0002
  SlotType = 0002
  TransmitFlowControl = 2
  VendorName = QLogic
  WWPN
  iScsiOffloadMode = 3

DCIM_NICView
  AutoNegotiation = 2
  BusNumber = 3
  ControllerBIOSVersion = N/A
  CurrentMACAddress = 14:FE:B5:1A:40:62
  DataBusWidth = 000B
  DeviceNumber = 0
  EFIVersion = 7.2.19
  FCoEOffloadMode = 3
  FCoEWWNN
  FQDD = NIC.Mezzanine.2B-1-1
  FamilyVersion = 7.2.14
  FunctionNumber = 0
  InstanceID = NIC.Mezzanine.2B-1-1
  LastSystemInventoryTime = 20120606133232.000000+000
  LastUpdateTime = 20120606133219.000000+000
  LinkDuplex = 0
  LinkSpeed = 0
  MaxBandwidth = 100
  MediaType = 2
  MediaType = 3
  MinBandwidth = 0
  NicMode = 2
  PCIDeviceID = 16AE
  PCISubDeviceID = 1007
  PCISubVendorID = 14e4
  PCIVendorID = 14e4
  PermanentFCOEMACAddress = 00:10:18:d6:e7:a1
  PermanentMACAddress = 00:10:18:D6:E7:A0
  PermanentiSCSIMACAddress = 00:10:18:D6:E7:A1
  ProductName = Broadcom NetXtreme II 10 Gb Ethernet BCM57810 -
14:FE:B5:1A:40:62
```

```
  ReceiveFlowControl = 3
  SlotLength = 0004
  SlotType = 00B5
  TransmitFlowControl = 3
  VendorName = Broadcom Corp
  WWPN = 20:01:00:10:18:D6:E7:A1
  iScsiOffloadMode = 3

...
```

The output is trimmed to show two `DCIM_NICView` instances, the first partition on the first port of a Qlogic QMD8262 network daughter card (NDC) CNA controller and the first partition on the first port of a Broadcom BCM57810 mezzanine card on fabric B.

There will be one `DCIM_NICView` instance for each port of a non-partitioned controller and one `DCIM_NICView` instance for each partition of a partitionable controller. Once the target ports or partitions are identified by their `InstanceID`, they can be inventoried further.

All attributes of all ports and partitions can be obtained by enumerating the `DCIM_NICAttribute` class. This can be done as follows:-

```
winrm enumerate "cimv2/root/dcim/DCIM_NICAttribute" -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic
```

To obtain all attributes of a specific CNA port or partition, a filter can be applied as follows:-

```
winrm enumerate "cimv2/root/dcim/DCIM_NICAttribute" -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic -
dialect:http://schemas.dmtf.org/wbem/cql/1/dsp0202.pdf -filter:"select
* from DCIM_NICAttribute where FQDD='NIC.Mezzanine.2B-1-1'"

DCIM_NICString
  AttributeDisplayName = Chip Type
  AttributeName = ChipMdl
  CurrentValue = BCM57810 B0
  Dependency
  FQDD = NIC.Mezzanine.2B-1-1
  GroupDisplayName = Broadcom Main Configuration Page
  GroupID = VndrConfigPage
  InstanceID = NIC.Mezzanine.2B-1-1:ChipMdl
  IsReadOnly = true
  MaxLength = 0
  MinLength = 0
  PendingValue
  ValueExpression


DCIM_NICString
  AttributeDisplayName = PCI Device ID
  AttributeName = PCIDeviceID
  CurrentValue = 168E
  Dependency
  FQDD = NIC.Mezzanine.2B-1-1
  GroupDisplayName = Broadcom Main Configuration Page
  GroupID = VndrConfigPage
  InstanceID = NIC.Mezzanine.2B-1-1:PCIDeviceID
  IsReadOnly = true
  MaxLength = 0
  MinLength = 0
  PendingValue
  ValueExpression
```

All instances of `DCIM_NICString`, `DCIM_NICInteger` and `DCIM_NICEnumeration` for the specified port or partition will be returned.

Using a combination of data returned by enumerating `DCIM_NICView`, `DCIM_NICString`, `DCIM_NICInteger` and `DCIM_NICEnumeration` (the latter three collectively grouped in `DCIM_NICAttribute`), the inventory of all CNA devices can be obtained and evaluated for configurability.

NIC Partitioning Enablement

On certain CNA controllers, NIC Partitioning can be turned off. This means the controller behaves like a traditional NIC device with only individual ports being available for various operations. This section describes how to turn on NIC partitioning if it is turned off.

1. Prior to turning on partitioning, we first need to identify if NIC Partitioning is supported by the controller. This can be done by getting the `DCIM_NICCapabilities` class for the instance of interest and checking the value of the `NicPartitioningSupport` property. The numeric values below are defined in the MOF file for the `DCIM_NICCapabilities` class and are mapped as follows:

   0 = Unknown
   2 = Supported
   3 = Not Supported

   ```
   winrm get
   "cimv2/root/dcim/DCIM_NICCapabilities?InstanceID=NIC.Mezzanine.2B-1-1"
   -r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
   encoding:utf-8 -a:basic -format:pretty
   ```

   ```
   DCIM_NICCapabilities
     BPESupport = 3
     CongestionNotification = 3
     DCBExchangeProtocol = 2
     ETS = 2
     EVBModesSupport = 3
     EnergyEfficientEthernet = 3
     FCoEBootSupport = 2
     FCoEMaxIOsPerSession = 0
     FCoEMaxNPIVPerPort = 0
   ```

```
  FCoEMaxNumberExchanges = 0
  FCoEMaxNumberLogins = 0
  FCoEMaxNumberOfFCTargets = 0
  FCoEMaxNumberOutStandingCommands = 0
  FCoEOffloadSupport = 2
  FQDD = NIC.Mezzanine.2B-1-1
  FeatureLicensingSupport = 3
  FlexAddressingSupport = 2
  IPSecOffloadSupport = 3
  InstanceID = NIC.Mezzanine.2B-1-1
  MACSecSupport = 3
  NWManagementPassThrough = 3
  NicPartitioningSupport = 2
  OSBMCManagementPassThrough = 3
  OnChipThermalSensor = 2
  OpenFlowSupport = 3
  PXEBootSupport = 2
  PartitionWOLSupport = 2
  PriorityFlowControl = 2
  RDMASupport = 3
  RXFlowControl = 3
  RemotePHY = 3
  TCPChimneySupport = 2
  TXBandwidthControlMaximum = 2
  TXBandwidthControlMinimum = 2
  TXFlowControl = 3
  VEBVEPAMultiChannel = 3
  VEBVEPASingleChannel = 3
  VFSRIOVSupport = 2
  VirtualLinkControl = 2
  WOLSupport = 2
  iSCSIBootSupport = 2
  iSCSIOffloadSupport = 2
  uEFISupport = 2
```

From the response, we notice that this controller does support partitioning.

`NicPartitioningSupport` is also available by enumerating the `DCIM_NICString` class as follows:-

```
winrm get "cimv2/root/dcim/DCIM_NICString?InstanceID=NIC.Mezzanine.2B-
1-1:NicPartitioningSupport" -r:https://10.0.0.1/wsman -u:root -p:******
-SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -format:pretty
```

```
DCIM_NICString
  AttributeDisplayName = NPS
  AttributeName = NicPartitioningSupport
  CurrentValue = Available
  Dependency
  FQDD = NIC.Mezzanine.2B-1-1
  GroupDisplayName = Broadcom Main Configuration Page
  GroupID = VndrConfigPage
  InstanceID = NIC.Mezzanine.2B-1-1:NicPartitioningSupport
  IsReadOnly = true
  MaxLength = 12
  MinLength = 0
  PendingValue
  ValueExpression
```

2. Next, let's go ahead and check the current state of NIC Partitioning – is it already enabled or disabled. This can be done by getting the `NicPartitioning DCIM_NICEnumeration` attribute.

   ```
   winrm get
   "cimv2/root/dcim/DCIM_NICEnumeration?InstanceID=NIC.Mezzanine.2B-1-
   1:NicPartitioning" -r:https://10.0.0.1/wsman -u:root -p:****** -
   SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -format:pretty
   ```

   ```
   DCIM_NICEnumeration
     AttributeDisplayName = NIC Partition
     AttributeName = NicPartitioning
     CurrentValue = Disabled
     Dependency
     FQDD = NIC.Mezzanine.2B-1-1
     GroupDisplayName = Device Configuration
     GroupID = DeviceLevelConfig
     InstanceID = NIC.Mezzanine.2B-1-1:NicPartitioning
     IsReadOnly = false
     PendingValue
     PossibleValues = Disabled
     PossibleValues = Enabled
     PossibleValuesDescription = Disabled
     PossibleValuesDescription = Enabled
   ```

   We notice that it is currently disabled. Notice also that it is possible to enable `NicPartitioning` since `Enabled` is a valid possible value for the attribute.

3. In order to enable partitioning, we can use the `SetAttribute()` or `SetAttributes()` methods on the `DCIM_NICService` class. Since there is only one attribute of interest at this time, we shall use the former method.

   ```
   winrm invoke SetAttribute
   "cimv2/root/dcim/DCIM_NICService?SystemCreationClassName=DCIM_ComputerS
   ystem+CreationClassName=DCIM_NICService+SystemName=DCIM:ComputerSystem+
   Name=DCIM:NICService"
   @{AttributeName="NicPartitioning";AttributeValue="Enabled";Target="NIC.
   Mezzanine.2B-1-1"}
   -r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
   encoding:utf-8 -a:basic -format:pretty

   SetAttribute_OUTPUT
     Message = The command was successful
     MessageID = NIC001
     RebootRequired = Yes
     ReturnValue = 0
     SetResult = Set PendingValue
   ```

Notice the syntax of the winrm command line. The `SetAttribute()` method is invoked with the `AttributeName`, `AttributeValue` and `Target` parameters. The target parameter equates to the `InstanceID` of the `DCIM_NICView` of interest. Details of using this command are documented in the Simple NIC Profile.

4. Setting a value marks the attribute in the pending state. We can verify that the attribute has a pending value by retrying step 2.

```
winrm get
"cimv2/root/dcim/DCIM_NICEnumeration?InstanceID=NIC.Mezzanine.2B-1-
1:NicPartitioning" -r:https://10.0.0.1/wsman -u:root -p:****** -
SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -format:pretty
```

```
DCIM_NICEnumeration
  AttributeDisplayName = NIC Partition
  AttributeName = NicPartitioning
  CurrentValue = Disabled
  Dependency
  FQDD = NIC.Mezzanine.2B-1-1
  GroupDisplayName = Device Configuration
  GroupID = DeviceLevelConfig
  InstanceID = NIC.Mezzanine.2B-1-1:NicPartitioning
  IsReadOnly = false
  PendingValue = Enabled
  PossibleValues = Disabled
  PossibleValues = Enabled
  PossibleValuesDescription = Disabled
  PossibleValuesDescription = Enabled
```

Note that a pending value is noted in the output.

5. In order to commit the change to the CNA controller, we need to create a Dell Lifecycle Controller job. Since the commit requires the machine to reboot and execute System Services, an immediate response for the WS-MAN call is not possible. Instead, a job ID is provided that can be used to track the progress of the commit operation.

   A job can be created by executing the `CreateTargetedConfigJob()` method on the `DCIM_NICService` on the `InstanceID` of the `DCIM_NICView` of interest.

```
winrm invoke CreateTargetedConfigJob
"cimv2/root/dcim/DCIM_NICService?SystemCreationClassName=DCIM_ComputerS
ystem+CreationClassName=DCIM_NICService+SystemName=DCIM:ComputerSystem+
Name=DCIM:NICService"
@{RebootJobType="3";ScheduledStartTime="TIME_NOW";Target="NIC.Mezzanine
.2B-1-1"} -r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -
SkipCAcheck -encoding:utf-8 -a:basic -format:pretty
```

```
CreateTargetedConfigJob_OUTPUT
  Job
    EndpointReference
      Address =
http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      ReferenceParameters
```

```
        ResourceURI = http://schemas.dell.com/wbem/wscim/1/cim-
schema/2/DCIM_LifecycleJob
        SelectorSet
          InstanceID = JID_389891129323
          __cimnamespace = root/dcim
  ReturnValue = 4096
```

Notice the syntax of the winrm command line. The `CreateTargetedConfigJob()` method is invoked with the `Target, RebootJobType` and `ScheduledStartTime` parameters. The `Target` parameter equates to the `InstanceID` of the `DCIM_NICView` of interest. The `ScheduledStartTime` parameter can be used to schedule a job at a future time. We are using the `TIME_NOW` value to kick off the job on the next reboot. The `RebootJobType` parameter is used to reboot the system right away. This parameter can be skipped if the job is to be executed at a future time. Details of using this command are documented in the Simple NIC Profile.

6. The command provides us with the `InstanceID` of the job created to execute this command. It can be polled using the `DCIM_LifecycleJob` class. Details working jobs are documented in the Job Control Profile.

```
winrm get
"cimv2/root/dcim/DCIM_LifecycleJob?InstanceID=JID_389891129323" -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic -format:pretty
```

```
DCIM_LifecycleJob
  ElapsedTimeSinceCompletion = 10
  InstanceID = JID_389891129323
  JobStartTime = TIME_NOW
  JobStatus = Completed
  JobUntilTime = TIME_NA
  Message = Job completed successfully
  MessageArguments = NA
  MessageID = PR19
  Name = ConfigNIC:NIC.Mezzanine.2B-1-1
  PercentComplete = 100
```

The `JobStatus` will go through the following states before completion: `New, Scheduled, Running, Completed`. In case of errors, states such as `Completed with Errors` or `Failed` are also possible.

7. Repeating step 2, we can verify that the value of `NicPartitioning` has been updated as needed.

```
winrm get
"cimv2/root/dcim/DCIM_NICEnumeration?InstanceID=NIC.Mezzanine.2B-1-
1:NicPartitioning" -r:https://10.0.0.1/wsman -u:root -p:****** -
SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -format:pretty
```

```
DCIM_NICEnumeration
  AttributeDisplayName = NIC Partition
  AttributeName = NicPartitioning
  CurrentValue = Enabled
```

```
    Dependency
    FQDD = NIC.Mezzanine.2B-1-1
    GroupDisplayName = Device Configuration
    GroupID = DeviceLevelConfig
    InstanceID = NIC.Mezzanine.2B-1-1:NicPartitioning
    IsReadOnly = false
    PendingValue
    PossibleValues = Disabled
    PossibleValues = Enabled
    PossibleValuesDescription = Disabled
    PossibleValuesDescription = Enabled
```

Now that `NicPartitioning` is enabled, we should see 8 `DCIM_NICView` instances for this controller, 4 partitions per port for a 2-port device with the following `InstanceIDs`.

```
winrm enumerate "cimv2/root/dcim/DCIM_NICView" -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic -format:pretty -
dialect:http://schemas.dmtf.org/wbem/cql/1/dsp0202.pdf -filter:"select
InstanceID from DCIM_NICView"

...

DCIM_NICView
  InstanceID = NIC.Mezzanine.2B-1-1

DCIM_NICView
  InstanceID = NIC.Mezzanine.2B-1-2

DCIM_NICView
  InstanceID = NIC.Mezzanine.2B-1-3

DCIM_NICView
  InstanceID = NIC.Mezzanine.2B-1-4

DCIM_NICView
  InstanceID = NIC.Mezzanine.2B-2-1

DCIM_NICView
  InstanceID = NIC.Mezzanine.2B-2-2

DCIM_NICView
  InstanceID = NIC.Mezzanine.2B-2-3

DCIM_NICView
  InstanceID = NIC.Mezzanine.2B-2-4
```

Note that `NicPartitioning` is a device-level configuration parameter. Even though it does show up under `NIC.Mezzanine.2B-1-1` as well as `NIC.Mezzanine.2B-2-1` (both ports), it only needs to be enabled in one place for the effect to apply across the entire controller.

Also, `NIC.Mezzanine.2B-1-1` points to the port when NIC partitioning is turned off, and to the first partition when partitioning is turned on. Both entities share the same PCI bus:device:function and hence are functionally equivalent. All port-level attributes that show up under the port when partitioning is turned off can be found on the first partition when partitioning is turned on.

Personality Enablement

CNA controllers are capable of multiple protocols or personalities. This section describes how to identify the personality capabilities of a CNA and to configure them as required for the server configuration.

1. For starters, we need to identify what personalities a CNA is capable of. This is available in the `DCIM_NICCapabilities` class and can be checked to show what personalities are supported. This class can be enumerated similar to step 1 in the NIC Partitioning Enablement section covered earlier.

   The following properties are of interest:-

   ```
   iSCSIOffloadSupport = 2

   FCoEOffloadSupport = 2
   ```

   If `iSCSIOffloadSupport` is available (2) then the iSCSI personality can be enabled on the ports or partitions. Likewise for `FCoEOffloadSupport`. Per the earlier enumeration, we can see that `NIC.Mezzanine.2B-1-1` is capable of both personalities.

   NIC personality is available by default on CNA controllers.

2. The next step is to identify which ports or partitions allow personality changes and their current settings. These are represented using the following attributes:-

   - NicMode
   - iScsiOffloadMode
   - FCoEOffloadMode

   The current state of these attributes can be obtained quickly in the output of the `DCIM_NICView` class, described previously in the NIC Inventory section. The states of the above three mentioned attributes is available as properties in the output. The numeric values are defined in the MOF file for the `DCIM_NICView` class and are mapped as follows:-

   0 = Unknown
   2 = Enabled
   3 = Disabled

   The capability of an individual port or partition is not evident in the `DCIM_NICView` output since it only captures the current state of the port or partition. The presence or absence of above `DCIM_NICEnumeration` attributes on a particular port or partition can be used to confirm whether that specific personality is supported or not on that port or partition since they are populated based on the controller's capabilities.

   ```
   winrm get
   "cimv2/root/dcim/DCIM_NICEnumeration?InstanceID=NIC.Mezzanine.2B-1-
   ```

```
1:NicMode" -r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -
SkipCAcheck -encoding:utf-8 -a:basic -format:pretty


DCIM_NICEnumeration
  AttributeDisplayName = Ethernet Protocol
  AttributeName = NicMode
  CurrentValue = Enabled
  Dependency = <Dep><AttrLev Op="OR"><ROIf
Name="NicPartitioning">Disabled</ROIf></AttrLev></Dep>
  FQDD = NIC.Mezzanine.2B-1-1
  GroupDisplayName = Partition 1
  GroupID = ConfigureForm1
  InstanceID = NIC.Mezzanine.2B-1-1:NicMode
  IsReadOnly = false
  PendingValue
  PossibleValues = Disabled
  PossibleValues = Enabled
  PossibleValuesDescription = Disabled
  PossibleValuesDescription = Enabled
```

NicMode is Enabled on partition 1 port 1 of this controller and can be Disabled if so required.

```
winrm get
"cimv2/root/dcim/DCIM_NICEnumeration?InstanceID=NIC.Mezzanine.2B-1-
1:iScsiOffloadMode" -r:https://10.0.0.1/wsman -u:root -p:****** -
SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -format:pretty


DCIM_NICEnumeration
  AttributeDisplayName = iSCSI Offload Protocol
  AttributeName = iScsiOffloadMode
  CurrentValue = Disabled
  Dependency = <Dep><AttrLev Op="OR"><ROIf
Name="NicPartitioning">Disabled</ROIf></AttrLev></Dep>
  FQDD = NIC.Mezzanine.2B-1-1
  GroupDisplayName = Partition 1
  GroupID = ConfigureForm1
  InstanceID = NIC.Mezzanine.2B-1-1:iScsiOffloadMode
  IsReadOnly = false
  PendingValue
  PossibleValues = Disabled
  PossibleValues = Enabled
  PossibleValuesDescription = Disabled
  PossibleValuesDescription = Enabled

winrm get
"cimv2/root/dcim/DCIM_NICEnumeration?InstanceID=NIC.Mezzanine.2B-1-
1:FCoEOffloadMode" -r:https://10.0.0.1/wsman -u:root -p:****** -
SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -format:pretty
```

```
DCIM_NICEnumeration
  AttributeDisplayName = FCoE Offload Protocol
  AttributeName = FCoEOffloadMode
  CurrentValue = Disabled
  Dependency = <Dep><AttrLev Op="OR"><ROIf
Name="NicPartitioning">Disabled</ROIf></AttrLev></Dep>
  FQDD = NIC.Mezzanine.2B-1-1
  GroupDisplayName = Partition 1
  GroupID = ConfigureForm1
  InstanceID = NIC.Mezzanine.2B-1-1:FCoEOffloadMode
  IsReadOnly = false
  PendingValue
  PossibleValues = Disabled
  PossibleValues = Enabled
  PossibleValuesDescription = Disabled
  PossibleValuesDescription = Enabled
```

iScsiOffloadMode and FCoEOffloadMode are both Disabled and can be Enabled if required.

If we look for `FCoEOffloadMode` on `NIC.Integrated.1-1-1`, an error is returned since partition 1 doesn't support FCoE offload.

```
winrm get
"cimv2/root/dcim/DCIM_NICEnumeration?InstanceID=NIC.Integrated.1-1-
1:FCoEOffloadMode" -r:https://10.0.0.1/wsman -u:root -p:****** -
SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -format:pretty
```

```
<s:Fault xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
    <s:Code>
        <s:Value>s:Sender</s:Value>
        <s:Subcode>
            <s:Value>wsman:InvalidParameter</s:Value>
        </s:Subcode>
    </s:Code>
    <s:Reason>
        <s:Text xml:lang="en">CMPI_RC_ERR_INVALID_PARAMETER</s:Text>
    </s:Reason>
    <s:Detail>
<wsman:FaultDetail>http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDeta
il/MissingValues</wsman:FaultDetail>
    </s:Detail>
</s:Fault>

Error number:  -2144108486 0x8033803A
```

The WS-Management service cannot process the request because a parameter for the operation is not valid.

However, FCoE offload shows up on partition 4.

```
winrm get
"cimv2/root/dcim/DCIM_NICEnumeration?InstanceID=NIC.Integrated.1-1-
```

```
4:FCoEOffloadMode" -r:https://10.0.0.1/wsman -u:root -p:****** -
SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -format:pretty
```

```
DCIM_NICEnumeration
  AttributeDisplayName = FCoE Offload Mode
  AttributeName = FCoEOffloadMode
  CurrentValue = Disabled
  Dependency = <Dep><AttrLev Op="OR"><ROIf
Name="NicMode\133Partition\0724\135">Enabled</ROIf></AttrLev></Dep>
  FQDD = NIC.Integrated.1-1-4
  GroupDisplayName = PARTITION 4 CONFIGURATION
  GroupID = ConfigureForm4
  InstanceID = NIC.Integrated.1-1-4:FCoEOffloadMode
  IsReadOnly = false
  PendingValue
  PossibleValues = Enabled
  PossibleValues = Disabled
  PossibleValuesDescription = Enabled
  PossibleValuesDescription = Disabled
```

Using the above data, the personality capabilities and current settings of the individual ports and partitions can be gauged and configured as needed.

Configuring the personalities now follows the same steps of calls described from step 3 in the NIC Partitioning Enablement section covered earlier. Following is an example of configuring two personalities on `NIC.Mezzanine.2B-1-1` in a single `SetAttributes()` call.

```
winrm invoke SetAttributes
"cimv2/root/dcim/DCIM_NICService?SystemCreationClassName=DCIM_ComputerS
ystem+CreationClassName=DCIM_NICService+SystemName=DCIM:ComputerSystem+
Name=DCIM:NICService" -r:https://10.0.0.1/wsman -u:root -p:****** -
SkipCNcheck -SkipCAcheck -encoding:utf-8

-a:basic -format:pretty -file:input.xml
```

Since we are providing an array of values, usage of an input file is required. The contents of `input.xml` referenced in the above command-line are:-

```
<p:SetAttributes_INPUT
xmlns:p="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/root/dcim/DCIM_NICService">
  <p:AttributeName>NicMode</p:AttributeName>
  <p:AttributeName>iScsiOffloadMode</p:AttributeName>
  <p:AttributeValue>Disabled</p:AttributeValue>
  <p:AttributeValue>Enabled</p:AttributeValue>
  <p:Target>NIC.Mezzanine.2B-1-1</p:Target>
</p:SetAttributes_INPUT>

SetAttributes_OUTPUT
  Message = The command was successful
  MessageID = NIC001
  RebootRequired = Yes
  ReturnValue = 0
  SetResult = Set PendingValue
```

Both values are set in a single invocation and the call returns successfully specifying that a reboot is required to commit these values.

As mentioned earlier, committing the values will require a `CreateTargetedConfigJob()` method call, followed by polling the created job ID for completion.

Bandwidth Allocation

As discussed earlier, the bandwidth allocation capability of partitionable CNA devices allows an administrator to intelligently configure the performance of individual partitions with respect to the available port bandwidth. This section discusses inventorying and configuring bandwidth allocation across the partitions on a port.

1. The bandwidth allocations for a partition are represented using the following attributes:-
   - MaxBandwidth
   - MinBandwidth

   To obtain a quick snapshot of a port or partition's current bandwidth settings, the above two properties can be referenced in the `DCIM_NICView` class

   The current state of these attributes can be obtained quickly in the output of the `DCIM_NICView` class, described previously in the NIC Inventory section. The states of the above two attributes is available as properties in the output. The numeric values represent a percentage of the total bandwidth available on the port.

   The individual `DCIM_NICInteger` attributes can be obtained as below. Note the use of filtering to enumerate the attributes of interest.

```
winrm enumerate "cimv2/root/dcim/DCIM_NICInteger" -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic -format:pretty -
dialect:http://schemas.dmtf.org/wbem/cql/1/dsp0202.pdf -filter:"select
* from DCIM_NICInteger where AttributeName like '%Bandwidth' and
FQDD='NIC.Mezzanine.2B-1-1'"


DCIM_NICInteger
  AttributeDisplayName = Partition 1 Relative Bandwidth Weight
  AttributeName = MinBandwidth
  CurrentValue = 0
  Dependency = <Dep><AttrLev Op="OR"><ROIf
Name="NicPartitioning">Disabled</ROIf></AttrLev></Dep>
  FQDD = NIC.Mezzanine.2B-1-1
  GroupDisplayName = Global Bandwidth Allocation Menu
  GroupID = GlobalBandwidthAllocation
  InstanceID = NIC.Mezzanine.2B-1-1:MinBandwidth
  IsReadOnly = false
  LowerBound = 0
  PendingValue
  UpperBound = 100

DCIM_NICInteger
  AttributeDisplayName = Partition 1 Maximum Bandwidth
  AttributeName = MaxBandwidth
  CurrentValue = 100
  Dependency = <Dep><AttrLev Op="OR"><ROIf
Name="NicPartitioning">Disabled</ROIf></AttrLev></Dep>
  FQDD = NIC.Mezzanine.2B-1-1
  GroupDisplayName = Global Bandwidth Allocation Menu
  GroupID = GlobalBandwidthAllocation
  InstanceID = NIC.Mezzanine.2B-1-1:MaxBandwidth
  IsReadOnly = false
  LowerBound = 1
  PendingValue
  UpperBound = 100
```

2. Setting the values of `MinBandwidth` and `MaxBandwidth` follows the same steps covered earlier in the NIC Partitioning Enablement section. This includes using the `SetAttribute()` or `SetAttributes()` methods to set new values along with a `CreateTargetedConfigJob()` method invocation to commit the values.

3. Setting `MaxBandwidth` across a port is straight-forward. Any value between 0-100% can be allocated to a partition since it is relative to the total available bandwidth at any given moment. All partitions on a port can be assigned 100% as the `MaxBandwidth` without any issues and the controller will intelligently provide as much as possible.

   In comparison, `MinBandwidth` is an absolute value – a partition has to be guaranteed that amount of bandwidth at the very least. Since it is an absolute value, the sum of `MinBandwidth` across a port need to add up 100%. Any commits where the total sum of `MinBandwidth` across a port exceeds 100% will be rejected by the controller as an invalid configuration. This means that at any given moment, the sum needs to be <= 100%.

   Since each partition is a separate entity with a distinct `InstanceID`, it needs a separate Dell Lifecycle Controller job to make configuration changes to it. Individual jobs are not aware of changes being made in other jobs. As a result, it is the responsibility of the entity requesting the jobs to order them in a fashion that will ensure that sum of `MinBandwidth` across a port are <= 100%.

   This can be achieved by scheduling jobs such that `MinBandwidth` values are first reduced, and then increased. Essentially, the delta values need to be sorted, and jobs scheduled in increasing order.

   Consider the following example of MinBandwidth values across the port:

Partition	1	2	3	4
Current value	25	25	25	25

Target value	30	30	20	20
Delta	+5	+5	-5	-5

As per the delta values, jobs for partitions 3 and 4 need to be scheduled before partitions 1 and 2 in order to ensure that at the end of each job, the sum of MinBandwidth stay <= 100%.

A more complex example:

Partition	1	2	3	4
Current value	5	5	50	40
Target value	40	50	5	5
Delta	+35	+45	-45	-35

In this case, job for partition 3 comes first, next is partition 4, then partition 1 and last is partition 2.

4. Once the ordering of jobs is determined, they can be created and scheduled accordingly. There are two ways to do this.

   - `CreateTargetedConfigJob()` in the correct order

   In this case, the jobs need to be created in the order that they need to run. For the second example above, assuming the values are already set, the following four commands would be executed to create jobs:-

```
winrm invoke CreateTargetedConfigJob
"cimv2/root/dcim/DCIM_NICService?SystemCreationClassName=DCIM_ComputerS
ystem+CreationClassName=DCIM_NICService+SystemName=DCIM:ComputerSystem+
Name=DCIM:NICService"
@{ScheduledStartTime="TIME_NOW";Target="NIC.Mezzanine.2B-1-3"} -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic -format:pretty


winrm invoke CreateTargetedConfigJob
"cimv2/root/dcim/DCIM_NICService?SystemCreationClassName=DCIM_ComputerS
ystem+CreationClassName=DCIM_NICService+SystemName=DCIM:ComputerSystem+
Name=DCIM:NICService"
@{ScheduledStartTime="TIME_NOW";Target="NIC.Mezzanine.2B-1-4"} -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic -format:pretty


winrm invoke CreateTargetedConfigJob
"cimv2/root/dcim/DCIM_NICService?SystemCreationClassName=DCIM_ComputerS
ystem+CreationClassName=DCIM_NICService+SystemName=DCIM:ComputerSystem+
Name=DCIM:NICService"
@{ScheduledStartTime="TIME_NOW";Target="NIC.Mezzanine.2B-1-1"} -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic -format:pretty


winrm invoke CreateTargetedConfigJob
"cimv2/root/dcim/DCIM_NICService?SystemCreationClassName=DCIM_ComputerS
ystem+CreationClassName=DCIM_NICService+SystemName=DCIM:ComputerSystem+
Name=DCIM:NICService"
@{RebootJobType=3;ScheduledStartTime="TIME_NOW";Target="NIC.Mezzanine.2
B-1-2"} -r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -
SkipCAcheck -encoding:utf-8 -a:basic -format:pretty
```

Note that all four jobs are scheduled to run at `TIME_NOW` but only the fourth command specifies a `RebootJobType` which actually kicks off the reboot of the host.

Usage of SetupJobQueue()

Another method is to create four jobs as above, but without `RebootJobType` or `ScheduledStartTime` parameters and then order them in the correct order using the `SetupJobQueue()` method.

```
winrm invoke SetupJobQueue
"cimv2/root/dcim/DCIM_JobService?CreationClassName=DCIM_JobService+Name
=JobService+SystemName=Idrac+SystemCreationClassName=DCIM_ComputerSyste
m" -r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -
SkipCAcheck -encoding:utf-8 -a:basic -format:pretty -file:input.xml
```

Contents of input.xml are as follows:-

```
<p:SetupJobQueue_INPUT
xmlns:p="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/root/dcim/DCIM_JobService">
  <p:JobArray>JID_390285447996</p:JobArray>
  <p:JobArray>JID_390285447997</p:JobArray>
  <p:JobArray>JID_390285447998</p:JobArray>
  <p:JobArray>JID_390285447999</p:JobArray>
  <p:StartTimeInterval>TIME_NOW</p:StartTimeInterval>
</p:SetupJobQueue_INPUT>

SetupJobQueue_OUTPUT
  Message = The command was successful
  MessageID = SUP025
  ReturnValue = 0
```

Since there has been no request to reboot the host to execute the jobs, a reboot job can also be created and scheduled in the above `SetupJobQueue()` call. A reboot job can be created as follows:-

```
winrm invoke CreateRebootJob
"cimv2/root/dcim/DCIM_SoftwareInstallationService?CreationClassName=DCI
M_SoftwareInstallationService+SystemCreationClassName=DCIM_ComputerSyst
em+SystemName=IDRAC:ID+Name=SoftwareUpdate" @{RebootJobType="3"} -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic -format:pretty
```

```
CreateRebootJob_OUTPUT
  RebootJobID
    EndpointReference
      Address =
http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      ReferenceParameters
        ResourceURI = http://schemas.dell.com/wbem/wscim/1/cim-
schema/2/DCIM_LifecycleJob
        SelectorSet
          InstanceID = RID_390287914172
          __cimnamespace = root/dcim
  ReturnValue = 4096
```

I/O Identity

The IO Identity feature of CNA controllers allows a remote administrator to assign known MAC and WWN/WWPN identities to the ports and partitions. This section covers configuring these values as required.

1. There are currently multiple attributes of interest from an IO Identity perspective:-
    - NicMode
        - MacAddr / VirtMacAddr
    - iScsiOffloadMode
        - IscsiMacAddr / VirtIscsiMacAddr
    - FCoEOffloadMode
        - FIPMacAddr / VirtFIPMacAddr
        - WWN / VirtWWN
        - WWPN / VirtWWPN

    Depending on the capabilities of the CNA, as discussed in the personality related sections earlier, for each supported personality on a port or partition, the corresponding identity attributes are available.

The factory programmed default identities exposed by the `MacAddr`, `IscsiMacAddr`, etc. can be overridden by assigning different values to the corresponding virtual instances. By default, the virtual instance is the same as the factory programmed values.

2. Prior to assigning identities to the CNA on modular systems, the FlexAddress feature on the Chassis Management Controller needs to be disabled.

   In addition, the iDRAC on the host system needs to be configured to disable FlexAddress. The attribute in question is `VirtualAddressManagement` and can be enumerated in the `DCIM_LCEnumeration` class as follows:-

```
winrm enumerate "cimv2/root/dcim/DCIM_LCEnumeration" -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic -format:pretty -
dialect:http://schemas.dmtf.org/wbem/cql/1/dsp0202.pdf -filter:"select
* from DCIM_LCEnumeration where
AttributeName='VirtualAddressManagement'"
```

```
DCIM_LCEnumeration
  AttributeName = VirtualAddressManagement
  CurrentValue = FlexAddress
  DefaultValue = FlexAddress
  ElementName = LC.emb.1
  InstanceID =
LifecycleController.Embedded.1#LCAttributes.1#VirtualAddressManagement
  IsReadOnly = false
  PendingValue
  PossibleValues = FlexAddress
  PossibleValues = Console
```

`VirtualAddressManagement` needs to be assigned a value of `Console` to ensure that FlexAddress is disabled. This can be done using the `SetAttribute()` and `CreateConfigJob()` methods on the `DCIM_LCService` class. Details of working with the LC service are documented in the LC Management Profile.

```
winrm invoke SetAttribute
"cimv2/root/dcim/DCIM_LCService?SystemCreationClassName=DCIM_ComputerSy
stem+CreationClassName=DCIM_LCService+SystemName=DCIM:ComputerSystem+Na
me=DCIM:LCService"
@{AttributeName="VirtualAddressManagement";AttributeValue="Console"} -
r:https://10.0.0.1/wsman -u:root -p:****** -SkipCNcheck -SkipCAcheck -
encoding:utf-8 -a:basic -format:pretty
```

```
SetAttribute_OUTPUT
  Message = The command was successful
  MessageID = LC001
  RebootRequired = No
  ReturnValue = 0
  SetResult = Set CurrentValue
```

The value can now be committed as follows:-

```
winrm invoke CreateConfigJob
"cimv2/root/dcim/DCIM_LCService?SystemCreationClassName=DCIM_ComputerSy
stem+CreationClassName=DCIM_LCService+SystemName=DCIM:ComputerSystem+Na
me=DCIM:LCService" -r:https://10.0.0.1/wsman -u:root -p:****** -
SkipCNcheck -SkipCAcheck -encoding:utf-8 -a:basic -format:pretty


CreateConfigJob_OUTPUT
  Job
    EndpointReference
      Address =
http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      ReferenceParameters
        ResourceURI = http://schemas.dell.com/wbem/wscim/1/cim-
schema/2/DCIM_LifecycleJob
        SelectorSet
          InstanceID = JID_389891129323
          __cimnamespace = root/dcim
  ReturnValue = 4096
```

3. Setting and committing the IO Identity attribute values follows the same steps as discussed in earlier sections, achieved by using methods such as `SetAttribute`() or `SetAttributes`() and `CreateTargetedConfigJob`() on `the DCIM_NICService`.
4. Resetting these values to their defaults can be done remotely by assigning the factory programmed default identities back to the virtual instances. The same effect is achieved when the host undergoes an A/C power loss.

Conclusion

The aim of this whitepaper was to introduce the powerful capabilities of CNA devices and provide the basic concepts to configure these capabilities using the Dell Lifecycle Controller Remote Services API. The above configuration scenarios demonstrate the powerful capabilities of this API, allowing administrators to configure thousands of machines remotely to match their exact needs with minimal time and effort.

Learn more

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

Reference Profiles
http://en.community.dell.com/techcenter/systems-management/w/wiki/1906.dcim-library-profile.aspx

Reference MOFs
http://en.community.dell.com/techcenter/systems-management/w/wiki/1840.dcim-library-mof.aspx

Best Practices Guide
http://en.community.dell.com/techcenter/extras/m/white_papers/20066173.aspx
Associated Scripts:
http://en.community.dell.com/techcenter/extras/m/white_papers/20066178.aspx

Web Services Interface Guide for Windows
http://en.community.dell.com/techcenter/extras/m/white_papers/20066174.aspx
Associated scripts
http://en.community.dell.com/techcenter/extras/m/white_papers/20066179.aspx

Web Services Interface Guide for Linux
http://en.community.dell.com/techcenter/extras/m/white_papers/20066176.aspx
Associated scripts
http://en.community.dell.com/techcenter/extras/m/white_papers/20066181.aspx

WS-MAN command line for Windows (Winrm)
http://msdn.microsoft.com/en-us/library/windows/desktop/aa384291(v=VS.85).aspx

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

Scripting the Dell Lifecycle Controller
http://en.community.dell.com/techcenter/systems-management/w/wiki/scripting-the-dell-lifecycle-controller.aspx

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