# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CISCO UCS SERVERS</td>
<td>3</td>
</tr>
<tr>
<td>RADWARE’S ALTEON VIRTUAL APPLIANCE (VA™) FOR NFV</td>
<td>4</td>
</tr>
<tr>
<td>Carrier Challenges and the NFV Working Group</td>
<td>4</td>
</tr>
<tr>
<td>Alteon VA for NFV</td>
<td>4</td>
</tr>
<tr>
<td>Integration with Cloud Orchestration Systems</td>
<td>5</td>
</tr>
<tr>
<td>Leveraging SDN Application Benefits in NFV Environments</td>
<td>5</td>
</tr>
<tr>
<td>Advanced Carrier Functionality</td>
<td>6</td>
</tr>
<tr>
<td>Seamless Control Plane Interoperability</td>
<td>6</td>
</tr>
<tr>
<td>Business and Technical Benefits Summary</td>
<td>7</td>
</tr>
<tr>
<td>Models and Specifications</td>
<td>8</td>
</tr>
<tr>
<td>Installation Prerequisites</td>
<td>8</td>
</tr>
<tr>
<td>Software and Hardware</td>
<td>9</td>
</tr>
<tr>
<td>Installation and Configuration</td>
<td>9</td>
</tr>
<tr>
<td>Install Alteon NFV</td>
<td>9</td>
</tr>
<tr>
<td>Configure Alteon NFV for Performance</td>
<td>13</td>
</tr>
<tr>
<td>lstopo</td>
<td>13</td>
</tr>
<tr>
<td>numctl</td>
<td>14</td>
</tr>
<tr>
<td>virsh capabilities</td>
<td>15</td>
</tr>
<tr>
<td>Performance Customization Example</td>
<td>15</td>
</tr>
<tr>
<td>Configure the NICs</td>
<td>19</td>
</tr>
<tr>
<td>Starting the Alteon NFV VM</td>
<td>22</td>
</tr>
<tr>
<td>Appendix A—Useful Virsh Commands</td>
<td>23</td>
</tr>
<tr>
<td>Technical Support</td>
<td>24</td>
</tr>
</tbody>
</table>
Cisco UCS Servers

The Cisco Unified Computing System (UCS) is a next-generation data center platform that unites compute, network, storage access, and virtualization into a cohesive system designed to reduce total cost of ownership (TCO) and increase business agility. The system integrates a low-latency lossless 10 Gigabit Ethernet unified network fabric with enterprise-class x86-architecture servers. The system is an integrated, scalable, multi-chassis platform in which all resources participate in a unified management domain. The main system components include:

- **Compute** — The system is based on an entirely new class of computing systems that incorporates blade servers based on Intel Xeon 5500 Series processors. The blade servers offer patented Cisco Extended Memory Technology to support applications with large data sets and allow more virtual machines per server.

- **Network** — The system is integrated onto a low-latency lossless 10-Gbps unified network fabric. This network foundation consolidates what today are three separate networks: LANs, SANs, and high-performance computing networks. The unified fabric lowers costs by reducing the number of network adapters, switches, and cables, and by decreasing power and cooling requirements.

- **Virtualization** — The system unleashes the full potential of virtualization by enhancing the scalability, performance, and operational control of virtual environments. Cisco security, policy enforcement, and diagnostic features are now extended into virtualized environments to better support changing business and IT requirements.

- **Storage access** — The system provides consolidated access to both SAN storage and network attached storage (NAS) over the unified fabric. Unifying storage access means that the Cisco Unified Computing System can access storage over Ethernet, Fibre Channel, Fibre Channel over Ethernet (FCoE), and iSCSI, providing customers with choice and investment protection. In addition, administrators can pre-assign storage-access policies for system connectivity to storage resources, simplifying storage connectivity and management while helping increase productivity.

- **Management** — The system uniquely integrates all the system components, enabling the entire solution to be managed as a single entity through Cisco UCS Manager software. Cisco UCS Manager provides an intuitive graphical user interface (GUI), a command-line interface (CLI), and a robust application-programming interface (API) to manage all system configuration and operations. Cisco UCS Manager helps increase IT staff productivity, enabling storage, network, and server administrators to collaborate on defining service profiles for applications. Service profiles are logical representations of desired physical configurations and infrastructure policies. They help automate provisioning and increase business agility, allowing data center managers to provision resources in minutes instead of days.

Working as a single, cohesive system, these components unify technology in the data center. They represent a radical simplification in comparison to traditional systems, helping simplify
data center operations while reducing power and cooling requirements. The system amplifies IT agility for improved business outcomes. The Cisco Unified Computing System components includes fabric interconnects, blade server chassis, blade servers, fabric extenders, and network adapters.

**Radware’s Alteon Virtual Appliance (VA™) for NFV**

Radware’s Alteon Virtual Appliance (VA) for Network Function Virtualization (NFV) environments decouples ADC functions from dedicated underlying hardware, allowing a next-generation ADC on x86 commercially off-the-shelf (COTS) hardware. Delivering a scalable, ultra-high capacity of up to 160Gbps per instance and multi-Tbps per cluster, Alteon VA for NFV reduces TCO, simplifies network services deployment, allows capacity elasticity, and automates service lifecycle management.4408 running on OnDemand Switch™ VL.

**Carrier Challenges and the NFV Working Group**

Today, every network service and functionality that wireless and wireline carriers need to roll out requires purchasing, deployment, and configuration of several physical network components from several equipment vendors. This has led carriers and operators to seek solutions and standards to simplify network operation, increase its agility, reduce implementation time of new and advanced network functions, and ultimately reduce their TCO.

This is why carriers and operators around the globe are trying to reduce their CAPEX and OPEX by standardizing the virtualization of their network functions, enabling them to run on x86 COTS servers and combining them with automation services. As a result, carriers, operators, and leading network equipment vendors have launched a new initiative to standardize the virtualization of network functions and built through an ETSI working group to support it, called Network Function Virtualization (NFV).

**Alteon VA for NFV**

As an active contributor in both the NFV and Software Define Network (SDN) working groups, Radware has developed a holistic strategy to enable carrier, large enterprise, and e-commerce networks to become smarter, more programmable, more flexible, and more cost-effective through SDN transformation and NFV compliance.

An important building block in that strategy is the Alteon VA for NFV environments, the first NFV-compliant software-based ADC on the market. Alteon VA for NFV was designed from the ground up for high performance and scalability, offering capacities up to 160Gbps per single instance running on x86 COTS servers. The result is breakthrough price-performance ratios for ADCs from 10Gbps to 160Gbps.

Further, by leveraging Radware’s “Elastic Scale,” SDN application carriers can dynamically support multi terabit/sec scale-out through clustering of multiple Alteon VAs for NFV instances.
Being part of the NFV echo-system, Alteon VA for NFV also features complete integration with NFV-based infrastructure virtualization and orchestration frameworks (KVM, OpenStack).

Alteon VA for NFV provides the most efficient resource utilization on commercially available hypervisors by redesigning the virtualization approach of Alteon VA and incorporating some new technologies that accelerate its overall performance. For example:

- Alteon NFV bypasses the hypervisor’s virtual switch, providing direct and faster access to the physical NICs of the server.
- Alteon NFV uses a fast packet processing algorithm for x86 servers based on Intel's DPDK code.

Integration with Cloud Orchestration Systems

To ensure carriers can fully leverage the benefits of Alteon VA in an NFV environment, Radware enables automation of its entire life-cycle through integration with NFV-based infrastructure virtualization and orchestration frameworks such as OpenStack. The life-cycle management automation includes all operations from provisioning a new Alteon VA instance and configuring the ADC service, through maintenance of the service, scaling its capacity when needed, and eventually decommissioning the instance. The lifecycle management operations are already available out-of-the-box in the various OpenStack orchestration distributions and interoperate with Radware’s Alteon VA for NFV.

Leveraging SDN Application Benefits in NFV Environments

Alteon VA for NFV is an essential part of Radware’s SDN application framework that enables smarter deployments of various network services such as traffic steering solutions with greater scalability and programmability throughout the network, breaking existing network barriers and overcoming capacity limited areas in the network.

The combination of Alteon VA for NFV with Radware’s SDN applications significantly simplifies implementation of advanced and complex network services, allowing improved operational efficiency of network management alongside application changes. Not every new network service rollout needs to become a networking project. While SDN streamlines network operations, Radware’s SDN applications and Alteon VA for NFV streamline network service provisioning and operations, as illustrated below:
Advanced Carrier Functionality

By leveraging a multi-proxy architecture, Alteon VA for NFV unleashes a set of unique application delivery services tailored and adjusted to carrier networks and service infrastructure needs that streamline mobile service delivery. It provides state-of-the-art transparent traffic steering based on mobile payload, headers, AAA, and other policy enforcement interfaces, including header modification capabilities to support various mobile use cases. In addition, it load balances both the control and data plane protocols.

Seamless Control Plane Interoperability

Alteon VA for NFV features a protocol and API-agnostic control-plane plug-in component that allows the solution to seamlessly interoperate with virtually any policy enforced eco-system (such as HSS, PCRFs, and RADIUS and Diameter-based APIs). This enables operators to automate user-aware and network-aware traffic steering functionality.

By providing instant conversion of different APIs, and normalization of the same protocols implemented by different vendors, Radware's control-plane plug-in eliminates long and costly R&D cycles needed to align different protocols that are implemented, while enabling fast and simple roll-out of new network applications, and automated user aware and/or network-aware real-time policy enforcement.
Business and Technical Benefits Summary

Radware’s Alteon VA for NFV solution enables operators and carriers to transform NFV benefits to reality, by migrating ADC services implementation from a dedicated hardware appliances deployment model to Radware’s NFV-based deployment model:

• Greater cost reduction:
  ▪ The ADC function can now run on standardized hardware, without requiring specialized dedicated hardware purchase and implementation, even for very high capacity applications.
  ▪ As a result, organizations can purchase the same x86 COTS servers for several network functions and in higher quantities, reducing the cost of their network-server hardware.
  ▪ Cost saving is also achieved by maintaining a reduced device RMA stock for all type of network functions.
  ▪ By decoupling the network function software from the underlying hardware, carriers avoid HW vendor lock.

• Reduced Operational Complexity
  ▪ Significantly faster rollout of complex applications and reduced operational complexity by using SDN applications together with Radware’s vDirect for service provisioning automation and NFV based network functions.
  ▪ Full life-cycle processes automation through integration with NFV oriented orchestration solutions, such as OpenStack

• Dynamic Scalability
  ▪ Operators can seamlessly scale/add more instances for increased service capacity, using standard COTS servers together with an “Elastic Scale” SDN application.
  ▪ Eliminates forklift upgrades, dynamically using/releasing the resources only when needed.

• Delivering Advanced Carrier Functions
  ▪ High capacity steering functions needed to efficiently route traffic to different Value Added Services (VAS) or monetization engines.
  ▪ Flexible service customization using AppShape++ policy scripting.
  ▪ Automated user-aware and/or network-aware real-time policy enforcement solution with Alteon VA for NFV, through out-of-the-box interoperability with the carrier’s policy enforcement/AAA eco system.
### Models and Specifications

<table>
<thead>
<tr>
<th>Functionality/Type</th>
<th>Alteon VA for NFV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput licenses [Gbps]</td>
<td>80, 160</td>
</tr>
<tr>
<td>Memory requirements</td>
<td>2-128 GB RAM</td>
</tr>
<tr>
<td>CPU requirements</td>
<td>2-16 vCPUs</td>
</tr>
<tr>
<td>Physical network interfaces</td>
<td>8 data interfaces</td>
</tr>
<tr>
<td>Management interface</td>
<td>1 management interface</td>
</tr>
<tr>
<td>Supported platforms</td>
<td>KVM hypervisor, VMware</td>
</tr>
</tbody>
</table>

### Installation Prerequisites

Perform the following steps before installing and configuring the solution:

1. Enable VT-d in the server BIOS.
2. Install Ubuntu Server 14.04.x LTS x64 with all updates.
3. Install KVM with (as root):
   a. libvirt 1.2.2
   b. QEMU 1.2.2 (recommended QEMU 2.0.0)
4. Install Linux bridge (apt-get install bridge-utils) as root:
   - Create a bridge for the management interface:
     i. brctl addbr mgmt
     ii. brctl addif mgmt eth0
     (make sure that you choose the correct management interface for your system)
5. Configure the IP address on the management bridge interface. For example:
   ```
   vi /etc/network/interfaces:
   auto eth0      "use the port defined for management
   iface eth0 inet manual "use the port defined for management
   iface mgmt inet static
   address 10.174.106.13
   netmask 255.255.0.0
   gateway 10.174.1.1
   ```
bridge_ports eth0  << use the port defined for management
bridge_stp off
auto mgmt

Software and Hardware

The following is a list of the hardware and software tested to verify the interoperability of the presented solution:

- Cisco UCS C240 M3S with 128GB RAM and dual Xeon E5-2690 v2 (20 cores or 40 Hyper threads)
- Radware’s Alteon NFV version 30.1
- Hypervisor KVM 2.0
- Ubuntu 14.04.1 LTS x64 Server
- 4x Intel x520 2ports 10Gbit SFP+ network cards

Installation and Configuration

This section describes how to install and configure the solution.

Install Alteon NFV

To install Alteon NFV

1. Download the latest Alteon NFV software to the UCS server.
2. Extract the Alteon files: `tar -xzvf AlteonOS-30-1-0-0_kvm_xen_cloud.tgz`
3. Run the install: `./AlteonOS-30.1.0.0/install/x86_64/bin/deployer`
4. Select OK.
5. Select **Local** (local deployment).

6. Select **images**, the default location.

7. Enter a name for the Alteon NFV instance (in this example, **AlteonOS-30.1.0.0**).

8. Choose number of vCPUs 1 to 16 (for maximum performance choose 18 vCPUs: 16 will be used, but two additional vCPUs are needed to balance the system)
9. Choose the desired amount of memory, 32MB should be sufficient.

![Memory Configuration]

10. Select the ‘mgmt’ bridge for the management network.

![Management Bridge Selection]

11. Select pass-through.

![Traffic Interface Type Selection]
12. Select the network adapters to use for pass-through.

13. Select No.

14. Select all the vCPUs for Alteon.

15. Choose all vCPUs for the Alteon, leaving 0 for FastView and AppWall
16. Press OK to finish the installation

**Configure Alteon NFV for Performance**

This section describes how to optimize Alteon NFV performance.

**Notes:**
- For optimization of Layer 4 throughput, use Full core and not Hyper-threading.
- Prefer using NIC's and CPU cores on the same NUMA node (CPU Socket). Use tools like `numactl`, `lstopo` to identify to which NUMA node each NIC is connected.

**lstopo**

The following is an example using the `lstopo` command:
Graphical output of the `lstopo` command:

```
Graphical output of the lstopo command:
```

```
numctl

Sample output of the `numctl --hardware` command:

```
[root@scapa ~]# numctl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 9 20 21 22 23 24 25 26 27 28 29
node 0 size: 65327 MB
node 0 free: 63709 MB
node 1 cpus: 10 11 12 13 14 15 16 17 18 19 30 31 32 33 34 35 36 37 38 39
node 1 size: 49152 MB
node 1 free: 47846 MB
node distances:
  node 0 1
  0: 10 20
  1: 20 10
```
virth capabilities

Sample output of `virsh capabilities`:

```
<topology>
  <cells num='2'>
    <cell id='0'>
      <cpu id='0' socket_id='0' core_id='0' siblings='1,2,3'/>
      <cpu id='1' socket_id='0' core_id='1' siblings='4,5,6'/>
      <cpu id='2' socket_id='0' core_id='2' siblings='7,8,9'/>
      <cpu id='3' socket_id='0' core_id='3' siblings='10,11,12'/>
      ...<
cell>
    </cell>
    <cell id='1'>
      <cpu id='13' socket_id='1' core_id='0' siblings='10,11,12'/>
      <cpu id='14' socket_id='1' core_id='1' siblings='13,14,15'/>
      <cpu id='15' socket_id='1' core_id='2' siblings='16,17,18'/>
      <cpu id='16' socket_id='1' core_id='3' siblings='19,20,21'/>
      ...<
cell>
  </cells>
</topology>
```

**Performance Customization Example**

**Notes:** When customizing Alteon NFV, consider the following:

- Reserve and map the CPUs (cores) for the specific Alteon NFV.
- Check the NUMA mapping of the PCI NICs. Try to use the NICs that are attached to the CPU where Alteon NFV will run, and allocate cores from that CPU.
- If the NICs are connected to both CPUs, it better to use cores from both CPUs.

The hardware used in this example is Cisco UCS C240 M3S with 128GB RAM and dual Xeon E5-2690 v2 (20 cores or 40 Hyper threaded cores).

The example uses:

- 8x 10Gbit network NICs
- 4x Intel cards that support DPDK and have 4x 10Gbit ports, connected to Numa-0 (CPU-0)
- 4x Intel cards that support DPDK and have 4x 10Gbit ports, connected to Numa-1 (CPU-1)

**Note:** Ensure that the installed 1x network card is installed in the PCIe that belongs to Numa-0, and the other network card is installed in the PCIe slot that is connected to Numa-1 (the hardware vendor needs to supply the hardware topology of the PCIe slots).

After installing the Alteon NFV software, pin the cores and NICs to the Alteon NFV vm.

The example uses 16 cores (not hyper-threaded, for better performance) for the Alteon NFV. (16x cores is the maximum virtual cores that can currently be used on the Alteon NFV).
To use both E5-2650 CPUs on the Alteon NFV to get better performance

1. From the CLI, run the command `virsh capabilities`.
2. Search for the `<topology>` field, and look at how the CPUs are mapped.

The following are sample topology entries and an explanation of the fields:

```xml
<cells num='2'>   --- The server has Dual CPUs
<cell id='0'>    --- Numa 0 (CPU0)
<memory unit='KiB'>65832408</memory>    --- The physical memory that are installed on Numa 0 (CPU0)
<cpus num='20'>   --- The number of hyper-threads
<cpu id='0' socket_id='0' core_id='0' siblings='0,20'/>   ---
  o Cpu id='0' – Is the first core on Numa 0
  o socket_id='0' – Is the Numa number
  o siblings='0,20'/> – Hyper-threading cores are 0 and 20
```
<topology>
  <cells num='2'>
    <cell id='0'>
      <memory unit='KiB'>65832408</memory>
      <cpus num='20'>
        <cpu id='0' socket_id='0' core_id='0' siblings='0,20'/>
        <cpu id='1' socket_id='0' core_id='1' siblings='1,21'/>
        <cpu id='2' socket_id='0' core_id='2' siblings='2,22'/>
        <cpu id='3' socket_id='0' core_id='3' siblings='3,23'/>
        <cpu id='4' socket_id='0' core_id='4' siblings='4,24'/>
        <cpu id='5' socket_id='0' core_id='8' siblings='5,25'/>
        <cpu id='6' socket_id='0' core_id='9' siblings='6,26'/>
        <cpu id='7' socket_id='0' core_id='10' siblings='7,27'/>
        <cpu id='8' socket_id='0' core_id='11' siblings='8,28'/>
        <cpu id='9' socket_id='0' core_id='12' siblings='9,29'/>
        <cpu id='10' socket_id='0' core_id='0' siblings='0,20'/>
        <cpu id='11' socket_id='0' core_id='1' siblings='1,21'/>
        <cpu id='12' socket_id='0' core_id='2' siblings='2,22'/>
        <cpu id='13' socket_id='0' core_id='3' siblings='3,23'/>
        <cpu id='14' socket_id='0' core_id='4' siblings='4,24'/>
        <cpu id='15' socket_id='0' core_id='8' siblings='5,25'/>
        <cpu id='16' socket_id='0' core_id='9' siblings='6,26'/>
        <cpu id='17' socket_id='0' core_id='10' siblings='7,27'/>
        <cpu id='18' socket_id='0' core_id='11' siblings='8,28'/>
        <cpu id='19' socket_id='0' core_id='12' siblings='9,29'/>
      </cpus>
    </cell>
    <cell id='1'>
      <memory unit='KiB'>66057760</memory>
      <cpus num='20'>
        <cpu id='10' socket_id='1' core_id='0' siblings='10,30'/>
        <cpu id='11' socket_id='1' core_id='1' siblings='11,31'/>
        <cpu id='12' socket_id='1' core_id='2' siblings='12,32'/>
      </cpus>
    </cell>
  </cells>
</topology>
In this example, the cores that will be used are:

- **cpuset id 0 to 8** (the total is 9 cores: 8x cores for the SP engine, and 1x for the MP engine)
- **Numa-0 and 10 to 17** (the total is 9 cores: 8x cores for the SP engine and 1x for the MP engine) for Numa-1

3. Edit the Alteon NFV xml file with the `virsh edit AlteonOS-30-1-0-0-16-cores` command (**AlteonOS-30-1-0-0-16-cores** is the name of the VM).

4. Insert the `cputune` lines under the `<vcpu placement>` line. The following is an example of how this should now appear:

```xml
<domain type='kvm'>
  <name>AlteonOS-30-1-0-0-16-cores</name>
  <uuid>cde1bd7d-a474-4500-8d2e-5ca6c1bbbd6</uuid>
  <memory unit='KiB'>33554432</memory>
  <currentMemory unit='KiB'>33554432</currentMemory>
  <vcpu placement='static'>18</vcpu>
  <cputune>
    <vcpupin vcpu='0' cpuset='0'/>
  </cputune>
</domain>
```
<vcpupin vcpu='1' cpuset='1'/>
<vcpupin vcpu='2' cpuset='2'/>
<vcpupin vcpu='3' cpuset='3'/>
<vcpupin vcpu='4' cpuset='4'/>
<vcpupin vcpu='5' cpuset='5'/>
<vcpupin vcpu='6' cpuset='6'/>
<vcpupin vcpu='7' cpuset='7'/>
<vcpupin vcpu='8' cpuset='8'/>
<vcpupin vcpu='9' cpuset='10'/>
<vcpupin vcpu='10' cpuset='11'/>
<vcpupin vcpu='11' cpuset='12'/>
<vcpupin vcpu='12' cpuset='13'/>
<vcpupin vcpu='13' cpuset='14'/>
<vcpupin vcpu='14' cpuset='15'/>
<vcpupin vcpu='15' cpuset='16'/>
<vcpupin vcpu='16' cpuset='17'/>
<vcpupin vcpu='17' cpuset='18'/>
</cputune>

5. Save the xml file. **Caution:** Do not exit from the editor.

6. Because this configuration uses both E5-2690 CPUs, allocate memory for each set of vCPUs (0-8 and 9-17). In editing mode, find the line starting with the string `<feature policy=`, and under it insert the lines:

```
<cell cpus='0-8' memory='16777216'/>
<cell cpus='9-17' memory='16777216'/>
```

7. Save the xml file. **Caution:** Do not exit from the editor.

### Configure the NICs

To configure the NICs

1. Open a new console to the UCS server and run the command `lstopo`.

This displays the network card and NICs that are connected to the specific Numa. You can see there are:

- 4x 1GB On-board NICs that are connected to **Numa-0**
- 3x Intel network cards with 4x 10Gbit ports that are connected to **Numa-0**
- 1x Intel card with 4x 10Gbit ports that are connected to **Numa-1**
This configuration includes 1x Intel card (4x 10Gig ports) that is connected to Numa-0 and a second Intel card (4x 10Gig ports) that is connected to Numa-1.

The eth ports that are used are ETH20, ETH21, ETH22, and ETH23 for the first Intel card that is connected to Numa-0. The other four (4) ports that are used are ETH16, ETH18, ETH17, and ETH13 that are connected to Numa-1. See the following figure:

**Figure 2 — Istopo Display**

2. You now add the NICs to the Alteon NFV XML file. The lines you add include the PCI address and not the ETH number. To find the PCI address, use the `ethtool` command.

For example, run the command `ethtool -i eth20`, which produces the following output:

```
driver: ixgbe
version: 3.15.1-k
firmware-version: 0x80000208
**bus-info: 0000:0f:00.0**
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: no
```

Look for the line `bus-info: 0000:0f:00.0` (the PCI address for ETH20).
3. Repeat the `ethtool` command for each ETH interface that you selected for the Alteon NFV.

An example line that needs to be edited is:

```
<address domain='0x0000' bus='0xf' slot='0x00' function='0x0'/>
```

where:

- **Address domain='0x0000'** is the first four (4) numbers of the bus-info line as shown in the output of the `ethtool` command.
- **bus='0xf'** are the 5 and 6 numbers of the bus-info line as shown in the output of the `ethtool` command.
- **slot='0x00'** are 7 and 8 numbers of the bus-info line as shown in the output of the `ethtool` command.
- **function='0x0'** is the last number of the bus-info line as shown in the output of the `ethtool` command.

4. In the XML editor, add the NICs to the Alteon NFV XML file:

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0xf' slot='0x00' function='0x0'/>
  </source>
</hostdev>

<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0xf' slot='0x00' function='0x1'/>
  </source>
</hostdev>

<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x12' slot='0x00' function='0x0'/>
  </source>
</hostdev>

<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x12' slot='0x00' function='0x1'/>
  </source>
</hostdev>

<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x86' slot='0x00' function='0x0'/>
  </source>
</hostdev>
```
<hostdev>
  <hostdev mode='subsystem' type='pci' managed='yes'>
    <source>
      <address domain='0x0000' bus='0x86' slot='0x00' function='0x1'/>
    </source>
  </hostdev>
  <hostdev mode='subsystem' type='pci' managed='yes'>
    <source>
      <address domain='0x0000' bus='0x89' slot='0x00' function='0x0'/>
    </source>
  </hostdev>
  <hostdev mode='subsystem' type='pci' managed='yes'>
    <source>
      <address domain='0x0000' bus='0x89' slot='0x00' function='0x1'/>
    </source>
  </hostdev>
  <hostdev mode='subsystem' type='pci' managed='yes'>
    <source>
      <address domain='0x0000' bus='0x86' slot='0x00' function='0x1'/>
    </source>
  </hostdev>
</hostdev>

5. Save the XML file and exit.

**Starting the Alteon NFV VM**

This procedure describes how to start the Alteon NFV VM.

**To see the exact name of the VM**

1. Run the command `virsh list --all`. This displays all VMs that are installed on the KVM server. For example:

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>AlteonOS-30-1-0-0-16-cores</td>
<td>shut off</td>
</tr>
<tr>
<td>-</td>
<td>AlteonOS-30-1-0-0-4-cores</td>
<td>shut off</td>
</tr>
<tr>
<td>-</td>
<td>AlteonOS-30-1-0-0_rls_640</td>
<td>shut off</td>
</tr>
</tbody>
</table>

2. Select **AlteonOS-30-1-0-0-16-cores** and start it with the following command:

   `virsh start AlteonOS-30-1-0-0-16-cores`

3. To see if the Alteon VM is running, enter command `virsh list --all`. 
Appendix A—Useful virsh Commands

virsh edit <KVM name> - edit the XML of the VA-KVM

virsh list --all - list of all the available VMs

virsh define <*.xml> - define a new VM based on a XML file

virsh start <KVM name> - start the VM

virsh destroy <KVM name> - stop the VM

virsh undefine <KVM name> - to completely remove the VM (can't be un-done)

virsh --version - See virsh version

virsh console <KVM name> // ( to exit CTRL+] )

virsh autostart -- disable <KVM name> - After reboot of the host, this <KVM Name> will not be autostarted
Technical Support


For more information, your Radware Sales representative or:
U.S. and Americas: (866) 234-5763
International: +972(3) 766-8666